TUGAS AKHIR –TE 141599

PERANCANGAN KONTROL STABILITAS HEXAPOD ROBOT MENGGUNAKAN METODE *NEURO-FUZZY*

Muhammad Fajar Ramadhan
NRP 2215105038

Dosen Pembimbing
Ir. Rusdhianto Effendi, A.K., MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknik Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT –**TE 141599**

HEXAPOD ROBOT DESIGN STABILIZATION CONTROL USING NEURO – FUZZY METHOD

Muhammad Fajar Ramadhan
NRP 2215105038

Supervisor
Ir. Rusdhianto Effendi, A.K., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electro Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan "Perancangan Kontrol Stabilitas Hexapod Robot Menggunakan Metode *Neuro – Fuzzy*" adalah benar – benar hasil karya intelektual mandiri, di selesaikan tanpa menggunakan bahan – bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah di tulis secara lengkap pada daftar pustaka

Apabila pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku

Surabaya, Juli 2017

Muhammad Fajar Ramadhan
2215105038

-- Halaman ini sengaja dikosongkan --

PERANCANGAN KONTROL STABILITAS HEXAPOD ROBOT MENGGUNAKAN METODE *NEURO-FUZZY*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I

Ir. Rusdhianto Effendi A.K., M.T.
NIP. 1957 04 24 1985 02 1001

SURABAYA
JULI, 2017

PERANCANGAN KONTROL STABILITAS HEXAPOD ROBOT MENGGUNAKAN METODE *NEURO-FUZZY*

Muhammad Fajar Ramadhan – 2215105038

Ir. Rusdhianto Effendie A.K, MT. – 195704241985021001

ABSTRAK

Hexapod adalah robot *mobile* berkaki yang memiliki enam kaki dengan jumlah 3 kaki pada setiap sisinya. Pada penelitian ini akan melakukan aksi kontrol kestabilan pada robot hexapod sehingga dia mampu untuk menstabilkan tubuhnya ketika diam ataupun berjalan pada kondisi medan yang tidak rata dan bidang miring. Dengan menggunakan metode *neuro-fuzzy* yang merupakan perpaduan antara kecerdasan buatan *neural-network* dan logika *fuzzy logic* hexapod mampu memiliki respon yang lebih cepat sehingga pergerakan robot lebih halus dalam menstabilkan tubuhnya pada kondisi diam maupun berjalan. Penyelesaian dari persamaan kaki robot dapat menggunakan penyelesaian geometri dengan menggunakan *invers* kinematik dan *forward* kinematik, melakukan perhitungan posisi dan sudut antara tiap sendi robot dan dihitung posisi ujung kaki. Hasil pengujian menunjukkan bahwa dengan *gain input* sebesar masing-masing *roll* dan *pitch* 1/120 dan *output* sebesar 14 - 25 untuk tiap kaki, menunjukkan bahwa kontroler mampu menyeimbangkan kondisi tubuhnya sampai +/- 40 derajat, dengan pengujian masing-masing dilakukan untuk gangguan roll negatif dan positif dan pitch negatif dan positif, dengan nilai *error* pada gangguan *roll* positif sebesar (-0,3321 derajat), *error* gangguan *roll* negatif sebesar (0,1342 derajat), error gangguan *pitch* positif sebesar (0,5258 derajat) dan *error* gangguan *pitch* negatif sebesar (-0.1761 derajat).

Kata Kunci : Robot Hexapod 3-DOF, Kontrol Keseimbangan, *Neuro – Fuzzy, Invers Kinematics, Forward Kinematics*

-- Halaman ini sengaja dikosongkan --

HEXAPOD ROBOT DESIGN STABILIZATION CONTROL USING NEURO – FUZZY METHOD

Muhammad Fajar Ramadhan – 2215105038

Ir. Rusdhianto Effendie A.K, MT. – 195704241985021001

ABSTRACT

Hexapod is a legged mobile robot that has six legs with 3 feet on each side. In this study will perform stability control action on the hexapod robot so that he is able to stabilize his body when still or walking on uneven terrain conditions and incline. By using neuro-fuzzy method which is a combination of artificial neural-network intelligence and logic fuzzy logic hexapod able to have a faster response so that the movement of the robot is more smooth in stabilizing the body on the condition of silence or walking. Completion of the robot leg equations can use geometric settlement by using kinematic forward and invers, performing position and angle calculations between each robot joint and calculated foot position. The test results show that with the input gain of each roll and pitch 1/120 and the output of 14-25 for each leg, indicating that the controller is able to balance the condition of his body up to +/- 40 degrees, with each test performed for the roll disorder negative and positive pitch and negative and positive pitch, with error value on positive roll disturbance (-0.3321 degree), negative roll disturbance error (0,1342 degree), positive pitch error (0,5258 degree) and error a negative pitch disturbance of (-0.1761 degrees).

***Keyword** : Robot Hexapod, Stabilization Control, Neuro – Fuzzy, invers kinematics, forward kinematics*

-- Halaman ini sengaja dikosongkan --

KATA PENGANTAR

Dengan mengucapkan puji syukur atas kehadirat Allah SWT yang telah melimpahkan rahmat, hidayah serta karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul :

"PERANCANGAN KONTROL STABILITAS HEXAPOD ROBOT MENGGUNAKAN METODE *NEURO-FUZZY*"

Tugas Akhir ini merupakan sebagian syarat untuk menyelesaikan pendidikan Strata-I pada bidang Studi Teknik Sistem Pengaturan, Departemen Teknik Elektro, Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini penulis menyampaikan terima kasih sebesar-besarnya kepada :

1. Kedua orang tua, kakak, adik, dan teman-teman penulis atas limpahan doa, kasih sayang dan teladan hidup bagi penulis.
2. Bapak Rusdhianto Effendi, A.K., MT. selaku dosen pembimbing yang telah membimbing dari awal sampai terselesaikannya tugas akhir ini.
3. Seluruh staff pengajar dan administrasi Departemen Teknik Elektro FTE-ITS.
4. Semua pihak yang telah banyak membantu untuk menyelesaikan tugas akhir ini yang tidak dapat penulis sebutkan satu persatu.

Harapan saya sebagai penulis adalah semoga terselesaikannya Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu kedepannya. Sadar atas keterbatasan yang dimiliki oleh penulis karena hasil dari Tugas Akhir ini jauh dari kesempurnaan. Demikian penulis sudah berusaha semaksimal mungkin serta saran dan kritik penulis harapkan guna memperbaiki tugas akhir ini.

Surabaya, Juli 2017

Penulis

--Halaman ini sengaja dikosongkan--

DAFTAR ISI

-- Halaman ini sengaja dikosongkan --

DAFTAR GAMBAR

DAFTAR TABEL

-- Halaman ini sengaja dikosongkan --

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya teknologi yang semakin meningkat banyak dari manusia yang memanfaatkan alam untuk memenuhi kebutuhan hidup mereka, seperti contohnya dalam hal transportasi. Dalam hal transportasi atau mobilitas manusia selama ini sering bertumpu pada roda yang kemungkinan apabila digunakan untuk melewati medan yang *extreme* contohnya bebatuan dan semacamnya sering terjadi suatu masalah akibat ketidaksesuaian medan yang di lalui oleh roda tersebut[1]. Dalam hal ini para ilmuwan dan ahli teknologi berani mencontoh dan menerapkan keunikan sebagian hewan untuk dijadikan penyelesaian dari masalah ketidaksesuaian medan tersebut.

Pada penelitian kali ini adalah meniru sistem gerak pada serangga laba-laba yang lebih fleksibel dan lebih sesuai dengan medan yang tidak rata, kasar dan bergelombang dengan diterapkannya sistem pergerakan laba-laba pada robot di harapkan bisa mengurangi bahkan menyelesaikan permasalahan yang ada. Salah satu penerapanya menggunakan sistem robot berkaki 6 yang mampu bergerak sesuai dengan apa yang di perintahkan. Penggunaan kaki robot yang berjumlah 6 kaki tersebut juga memiliki masalah yaitu bagaimana menyelaraskan dan mengontrol antara kaki satu dengan lainnya agar dapat berjalan sesuai dengan medan yang di tempuhnya.

Terdapat beberapa metode yang digunakan untuk mengontrol pergerakan dari robot laba-laba ini salah satunya, yaitu : metode *neuro – fuzzy* metode ini menggabungkan antara jaringan saraf buatan dengan pengambilan keputusan dari logika *fuzzy logic*, dengan adanya penerapan metode ini pada robot laba-laba ini menjadikan pergerakan robot menjadi lebih halus dan *error positioning* robot menjadi lebih kecil[2].

1.2 Perumusan Masalah

Penerapan kontroler *neuro-fuzzy* dalam mengatur pergerakan robot agar posisi tubuh robot hexapod tetap stabil dengan posisi yang telah di berikan dalam keadaan diam maupun berjalan pada medan yang tidak rata dan bidang miring. Teknik kontrol pada setiap kaki robot agar selaras dalam penetuan posisi tubuh robot agar tetap stabil. Konversi dari *input* posisi pada sensor menjadi *output* pergerakan kaki – kaki robot sehingga mampu menstabilkan tubuhnya sendiri.

1.3 Batasan Masalah

Agar pengerjaan penelitian ini dapat terarah tanpa mengurangi maksud dan tujuannya, maka di tentukan batasan permasalahan sebagai berikut :

1. Melakukan pemodelan *forward* kinematik pada bagian kaki robot.
2. Melakukan pemodelan *invers* kinematik pada bagian kaki dan badan robot berbasis geometri.
3. Menggunakan robot *hexapod* 3 DOF pada masing-masing kaki.
4. Data dari IMU yang digunakan sebagai sensor kemiringan adalah data *gyroscope* dan *accelerometer*.
5. Perangkat kontroler yang digunakan adalah Arduino Mega 2560.
6. Menggunakan *neuro-fuzzy* sebagai metode kontrol untuk menyeimbangkan atau menstabilkan body robot.

1.4 Tujuan Penelitian

Pada penelitian di harapkan mampu mengontrol pergerakan dari *hexapod robot* dalam melewati medan yang sulit contohnya bidang yang tidak rata, bidang miring dan lain sebagainya, yang sulit di lalui oleh robot beroda dengan kondisi kestabilan tubuh robot yang di harapkan. Pergerakan robot menghasilkan pergerakan yang halus dan memiliki repson yang cepat dalam

menstabilkan tubuh robot pada saat kondisi diam maupun berjalan.

1.5 Metodologi

Metodologi yang digunakan dalam penyusunan penelitian ini adalah :

1. Studi Literatur
   Pada tahap ini, hal – hal yang akan dipelajari yaitu mengetahui cara kerja dan spesifikasi dari *hexapod* robot sebagai *plant* dari penelitian ini. Kemudian, dilanjutkan menentukan pergerakan robot dengan menganalisa kinematika robot. Kinematika robot diklasifikasikan menjadi 2 bagian yaitu *forward kinematisc* dan *inverse kinematics.* Setelah mempelajari kinematika robot maka selanjutnya mempelajari metode *neuro-fuzzy* yang akan digunakan untuk menyelesaikan perhitungan kinematika balik.

2. Menentukan Parameter DH
   Pada tahap ini, diperlukan untuk menentukan dan menghitung nilai parameter dari *hexapod*. Nilai parameter DH ini akan digunakan untuk merancang *forward kinematics*. Hasil dari *forward kinematics* adalah untuk mendapatkan posisi tujuan yang diinginkan dari *end effector* manipulator robot. Dari perancangan *forward kinematics*, posisi awal tersebut akan digunakan sebagai *input* dari *inverse kinematics*.

3. Penyelesaian Kinematika Balik dan Simulasi
   Setelah memperoleh hasil *forward kinematics* yang berupa posisi titik x,y,z dari *end effector* ujung kaki robot *hexapod* maka posisi awal yang akan dituju harus dihitung nilai sudut dari masing - masing *joint* robot *hexapod*. Untuk menghitung nilai sudut agar *end effector* mencapai posisi titik tersebut maka persamaan *inverse kinematics* harus diselesaikan. Penyelesaian permasalahan *inverse kinematics.*

4. Pengujian Kontroler *Neuro Fuzzy* dan Pengujian Penyelesaian *Invers* Kinematik
   Setelah melakukan penyelesaian *invers* kinematik selanjutnya adalah menyelesaikan perancangan kontroler untuk mestabilkan posisi tubuh robot hexapod dengan menggunakan logika *fuzzy* dan proses pembelajaran dari *neural network*. Kontroler ini akan menghasil posisi tubuh robot yang seharusnya jika tanah atau tempat berpijak kaki – kaki robot di miringkan pada sudut tertentu kemudian hasil posisi tadi akan diolah dengan penyelesaian *invers* kinematik untuk dicari posisi sudut yang sesuai dengan posisi keluaran dari kontroler.
5. Penulisan Buku Tugas Akhir
   Tahap penulisan buku Tugas Akhir terdiri dari pendahuluan, teori dasar, perancangan metode untuk menyelesaikan *inverse kinematics*, analisa dan hasil simulasi serta penutup yang berisikan kesimpulan dan saran untuk tugas akhir yang telah dikerjakan.

1.6 Sistematika Penulisan

Peneltian ini disusun dengan sistematika penulisan sebagai berikut :

BAB I : PENDAHULUAN

Bab ini berisi tentang pendahuluan yang terdiri dari latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, sistematika penulisan, serta relevansi dari penelitian ini.

BAB II : DASAR TEORI

Bab ini membahas mengenai teori-teori yang berkaitan dengan topik penelitian yang di dapatkan dari referensi buku, jurnal, datasheet, serta artikel yang lainnya. Pada penelitian ini akan dibahas yaitu invers kinematik forward kinematik, transformasi homogenus kontroler neuro – fuzzy sensor IMU (inertial Measuring Unit) dan spesifikasi hardware kontroler yang digunakan.

BAB III : PERANCANGAN SISTEM

Bab ini membahas tentang perancangan sistem yang meliputi perancangan mekanika robot terdiri dari kaki dan badan robot, perancangan forward kinematik dan invers kinematik. Penyelesaian logika fuzzy dengan jaringan saraf tiruan untuk menghitung posisi badan robot untuk menstabilkan posisi badan robot.

BAB IV : PENGUJIAN DAN ANALISA SISTEM

Bab ini membahas tentang seluruh hasil pengujian, analisa dari implementasi sistem yang dirancang pada penelitian ini. Pada penelitian ini akan dilakukan pengujian yaitu : pengujian motor servo yang digunakan dan menganalisanya, pengujian sensor IMU yang digunakan, pengujian penyelesaian invers kinematika dan pengujian kontroler dengan adanya gangguan berupa sudut kemiringan tanah tempat robot berpijak.

BAB V : PENUTUP

Bab ini berisi tentang kesimpulan hasil analisa dari implemetasi sistem dan penyempurnaan berupa saran dari implementasi yang telah dilakukan. Berupa kesimpulan dari pemakaian jenis motor servo yang tepat, proses proses pengambilan data sensor dan kontroler yang digunakan.

1.7 Relevansi

Hasil yang didapatkan dari penelitian ini diharapkan menjadi salah satu referensi yang digunakan oleh mahasiswa lain yang ingin menggunakan metode dan alur kerja pada buku ini.

-- Halaman ini sengaja dikosongkan --

BAB II
DASAR TEORI

Pada bagian ini akan dijelaskan mengenai materi – materi yang menjadi bagian dari landasan teori dan literatur mengenai penelitian robot berkaki yaitu tentang macam – macam konfgurasi robot berkaki, algoritma gerakan kaki, kinematika robot berkaki dan keseimbangan robot berkaki menggunakan metode kontrol logika *fuzzy* yang memiliki sistem pembelajaran *neural network (neuro-fuzzy)*.

2.1 Robot Hexapod

Hexapod robot merupakan robot yang menyerupai serangga berkaki 6 dengan 3 derajat kebebasan pada tiap kakinya, yang memungkinkan bergerak dengan fleksibel di berbagai medan. Keuntungan utama dari robot ini adalah stabilitas, tidak seperti robot bipedal robot ini stabil secara statis sehingga mereka tidak bergantung pada mekanisme keseimbangan[3]. Menurut bentuk tubuhnya *Hexapod* dibagi menjadi dua macam, yaitu : *rectangular* dan *hexagonal* atau *circular*. Tipe *rectangular* memiliki kaki tegak lurus terhadap titik sumbu kepala dan titik sumbu ekor (gambar 2.1a), sedangkan *hexagonal* kakinya mengelilingi seluruh badan robot (gambar 2.1b).

Gambar 2.1a *Rectangular*

Gambar 2.1b *Hexagonal*

2.1.1 Konfigurasi Robot Berkaki

Penggunaan robot berkaki didasarkan pada keunggulannya dibandingkan robot beroda yaitu bentuk berkaki dapat menjelajah rintangan atau landasan yang tidak rata dengan rintangan yang mana sebuah kendaraan beroda memiliki keterbatasan gerak, seperti pada lingkungan dengan kondisi bergelombang, berbatu, karang atau hutan. Robot berkaki banyak sekali di temukan seperti dua kaki

(*Humanoid*), empat kaki (*Quadruped*), enam kaki (*Hexapod*), dan konfigurasi delapan kaki (*Octopod*). Konfigurasi kaki robot yang di gunakan dalam penelitian kali ini adalah enam kaki (*Hexapod*).

Robot *hexapod* yang digunakan dalam peneltian ini memiliki bentuk tubuh yang *hexagonal* yaitu ke-enam kaki mengelilingi tubuhnya. Bentuk seperti ini sangat baik digunakan untuk berjalan memutar dan menyamping[3]. Robot berkaki enam (*hexapod*) memiliki tingkat keseimbangan yang tinggi sehingga cocok digunakan untuk judul penelitian kali ini. Ada dua tipe kestabilan robot yaitu kestabilan statis dan dinamis[3]. Kestabilan statis terjadi saat tidak dibutuhkannya gerakan atau gaya tambahan untuk mencegah badan robot terjatuh. Sedangkan kestabilan dinamis dibutuhkan ketika robot tidak dapat mempertahankan keseimbangan badannya tanpa memerlukan gaya atau gerakan tambahan. *Hexapod* memiliki kesatbilan statis yang lebih baik daripada *quadruped* karena jumlah kaki yang lebih banyak.

Kecepatan saat *hexapod* berjalan merupakan faktor yang paling penting dari sebuauh *mobile* robot. Beberapa riset menunjukkan bahwa kecepatan dari robot berkaki *Vn* adalah sebagai berikut,

$$Vn = \frac{Fs}{\beta T} + \frac{Fs}{tT}\left(\frac{1-\beta}{\beta}\right) \tag{2.1}$$

Dengan,

Fs = Panjang Langkah
T = *Cycle Time*
β = *Duty Factor*

Duty factor β secara langsung bergantung pada jumlah kaki dari robot. *Transfer phase (tT)* adalah waktu saat kaki berada di udara untuk membuat langkah berikutnya dan *support time (tS)* yaitu waktu saat kaki menyentuh tanah dan robot bergerak maju, *tT* dan *tS* adalah turunan dari *cycle time* dan *duty factor*.

$$tT = (1-\beta)T \tag{2.2}$$

$$tS = \beta\, T$$

Duty factor minimum dari *n* robot berkaki adalah $\beta n = 3/n$, sehingga kecepatan maksimal robot berkaki *n* dapat dihitung :

$$V4 = 0.333 \left(\frac{Fs}{\mathrm{tT}}\right) \tag{2.3}$$

$$V6 = \left(\frac{Fs}{\mathrm{tT}}\right) \tag{2.4}$$

$$V8 = 1.6 \left(\frac{Fs}{\mathrm{tT}}\right) \tag{2.5}$$

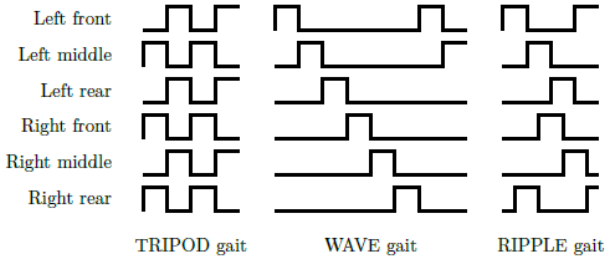
Dari persamaan diatas dapat dilihat bahwa *hexapod* memiliki kecepatan yang lebih tinggi dibandingkan *quadruped*, dan lebih rendah dari *octopod*[4].

2.1.2 Pola Langkah Robot *Hexapod* (Gait)

Pada robot berkaki berkaki enam *(hexapod)* memiliki banyak kombinasi dan desain penempatan kaki – kakinya, yaitu dapat dikonfugurasi misalnya langkah bentuk badan robot yang berbentuk persegi panjang *(rectangular)* dan berbentuk lingkaran *(hexagonal)*. Bentuk badan yang persegi panjang adalah desain yang lebih mirip bentuk binatang, desain ini cocok untuk gerakan maju akan tetapi kurang fleksibel untuk gerakan berputar dan menyamping[4].

Gait (gaya berjalan) sebuah robot berkaki enam jauh lebih banyak dibandingkan robot berkaki empat karena kombinasi pilihan untuk bergerak dengan kaki tunggal atau pasangan lebih besar. Serangga dapat berjalan dengan berbagai macam pola gerakan kaki pada kecepatan yang berbeda pula. Gerakan setiap kaki dapat dibagi menjadi *fase support*, dimana kaki memberikan dukungan dan mendorong badan robot dan *fase transfer* yaitu saat kaki diangkat dari tanah dan berayun maju untuk memulai fase lainnya.

Pada penelitian kali ini telah diamati beberapa contoh *gait* dari serangga dengan posisi kaki berayun seperti pada gambar 2.2. pada pola langkah tersebut merupakan pola langkah serangga pada kecepetan rendah yaitu dengan satu ayunan kaki tiap waktu sampai pola *tripod* berkecepatan tinggi dengan tiga pola langkah kaki yang saling bersilangan antara tiap sisi robot[5].



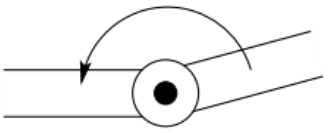
Gambar 2.2 Diagram *Gait Hexapod*

Bar hitam merupakan *fase support* dari kaki dan ruang antara bar merupakan *fase transfer*. Gaya berjalan yang ada pada gambar 2.2 yang paling atas adalah pola langkah yang lambat karena hanya satu kaki pada *fase transfer* setiap waktunya, berikutnya adalah *gait* yang cukup cepat dan terlihat alami. *Gait* ketiga adalah tipe *tripod gait* yang stabil statis, dan yang terakhir adalah kombinasi dari *tripod gait*.

2.2 Bentuk Sendi Robot

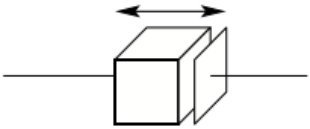
Manipulator robot adalah sistem mekanik yang menunjukkan pergerakan dari robot. Sistem mekanik ini terdiri dari susunan *link* (rangka) dan *joint* (engsel) yang mampu menghasilkan gerakan. Sendi pada robot berfungsi untuk menghubungkan dua *link*, antara *link* tersebut dapat menentukan arah putaran yang diinginkan sesuai dengan bentuk sendi dari manipulator robot. Secara umum sendi pada robot dapat didefinisikan menjadi dua bentuk, yaitu :

1. Sendi Putar (*Revolute Joint*)
   Revolute joint adalah sendi yang dapat bergerak secara berputar atau rotasi. Bentuk umum sendi putar dapat dilihat pada Gambar 2.3.



Gambar 2.3 Bentuk *Revolute Joint*

2. Sendi Geser (*Prismatic Joint*)
   Prismatic joint adalah sendi yang bergerak secara translasi atau disebut dengan sendi geser. Bentuk umum sendi geser dapat dilihat pada Gambar 2.4.

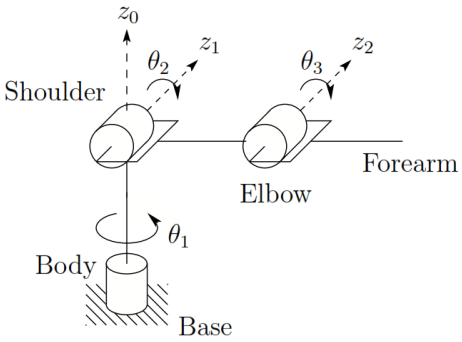


Gambar 2.4 Bentuk *Prismatic Joint*

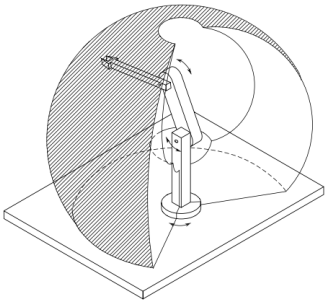
2.4 Konfigurasi Manipulator Robot

Secara umum struktur robot dapat dibedakan menurut sumbu koordinat yang digunakan, terdapat 5 klasifikasi robot yaitu :

1. *Articulated Configuration* (RRR)
   Articulated robot sering juga disebut dengan sendi putar (*revolute*). Struktur dari sendi *articulated* bersifat *revolute joint*. Konfigurasi *articulated* hampir memiliki pergerakan yang sama dengan pergerakan lengan manusia, dan hal ini memberikan fleksibilitas atau derajat kebebasan yang tinggi dalam mengakses objek. Pada Gambar 2.5 [1] dan 2.6 [4] merupakan struktur dan ruang kerja dari *articulated*.



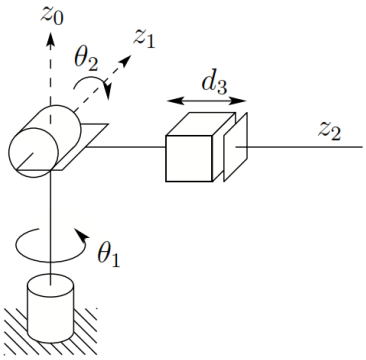
Gambar 2.5 Struktur dari *Articulated Configuration*



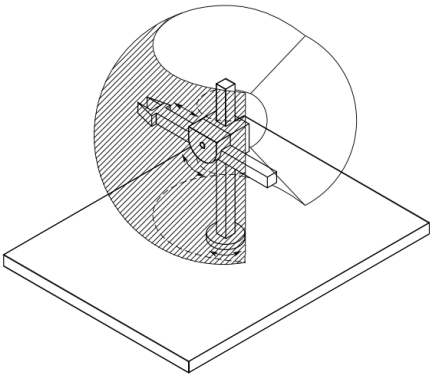
Gambar 2.6 Ruang Kerja dari *Articulated*

2. *Spherical Configuration* (RRP)

Konfigurasi *spherical* terdiri dari dua *revolute joint* dan satu *prismatic joint.* Daerah kerja dari *spherical* merupakan bagian bola berongga termasuk juga basis pendukungnya dari manipulator dan dengan demikian bisa memungkinkan manipulasi objek dilantai. Hal ini dapat dilihat pada Gambar 2.7 [1] dan 2.8 [4].



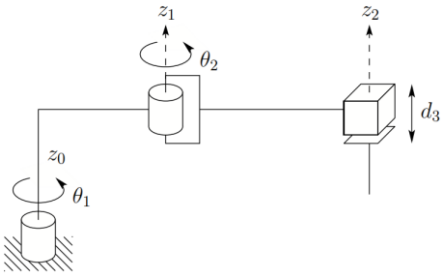
Gambar 2.7 Struktur dari *Spherical Configuration*



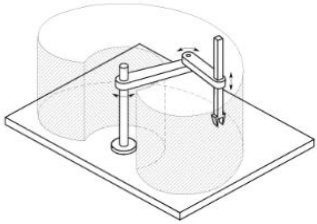
Gambar 2.8 Ruang Kerja dari *Spherical*

3. SCARA *Configuration* (RRP)

SCARA (*Selective Compliant Articulatedd Robot for Assembly*) konfigurasinya disesuaikan dengan operasi perakitan. Konfigurasi ini terdiri dari 2 *revolute* dan 1 *prismatic joint*. Konfigurasi SCARA didesain untuk memberikan pergerakan pada arah horizontal. Aplikasi robot SCARA dapat digunakan untuk pemindahan barang (assembly). Struktur dari SCARA dapat dilihat pada Gambar 2.9 [1] dan 2.10 [4].



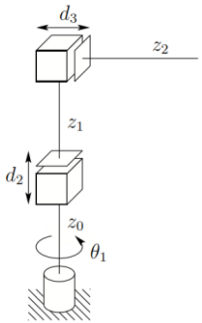
Gambar 2.9 Struktur dari SCARA *Configuration*



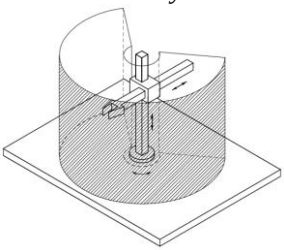
Gambar 2.10 Ruang Kerja dari SCARA

4. *Cylindrical Configuration* (RPP)

Konfigurasi *cylindrical* terdiri dari 1 *revolute* dan 2 *prismatic joint*. Konfigurasi ini memiliki kemampuan yaitu kecepatan pergerakan yang lebih tinggi pada bidang horizontal dibandingkan dengan sistem *cartesian* yang dapat dilihat pada Gambar 2.11[1] dan 2.12 [4]



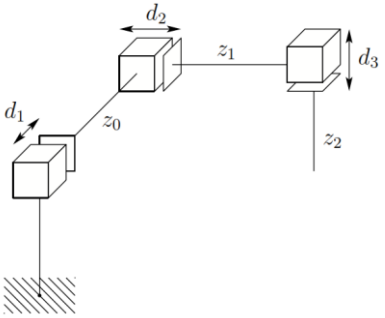
Gambar 2.11 Struktur dari *Cylindrical Configuration*



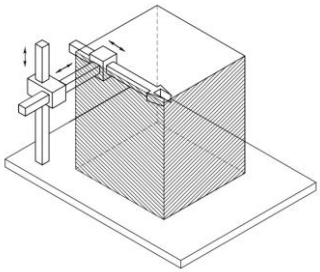
Gambar 2.12 Ruang Kerja dari *Cylindrical*

5. *Cartesian Configuration* (PPP)

Manipulator yang tiga *joint* pertamanya bersifat *prismatic*. *Cartesian* memiliki struktur konfigurasi yang paling kaku. Hal ini sangat menguntungkan untuk mengangkat beban yang berat dan pengulangan yang tinggi pada seluruh area pergerakan. Pada Gambar 2.13 [1] dan 2.14 [4] merupakan struktur dan daerah kerja dari *cartesian*.



Gambar 2.13 Struktur dari *Cartesian Configuration*



Gambar 2.14 Ruang Kerja dari Cartesian

2.3 Kinematika Robot

Kinematika robot adalah suatu bentuk deskripsi geometri matematik dari sebuah robot yang menerangkan antara konsep geometri ruang sendi dengan konsep koordinat yang digunakan untuk menentukan posisi dari suatu objek dilihat dari ruang 3 dimensi.

Dengan model kinematik, kita dapat menentukan konfigurasi referensi input yang harus diberikan pada setiap aktuator agar robot melakukan gerak yang simultan untuk mencapai posisi yang dikehendaki, sebagai contohnya input berupa posisi pada sumbu x, y dan z kemudian di modelkan dengan kinematika ini untuk menentukan berapa besar sudut yang harus dicapai untuk bisa berada pada posisi yang diinginkan, ini disebut *invers* kinematik. Sebaliknya jika input yang diberikan ke model kinematik adalah berupa sudut maka akan di ketahui posisi (x, y dan z) robot, ini disebut *forward* kinematik.

Jika robot *n*-DOF dengan tugas dinyatakan sebagai suatu fungsi trayektori atau titik – titik jalur P(t) dan posisi tiap sendi dinyatakan sebagai q(t) = (r,θ(t)), maka kinematik majunya dapat di ekspresikan sebagai :

$$P(t) = f(q(t)) \tag{2.6}$$

dengan,

$$q(t) = \begin{pmatrix} q1 \\ q2 \\ : \\ : \\ qn \end{pmatrix}$$

Dengan demikian kinematik baliknya adalah sebagai berikut :

$$P(t) = f^{-1}(q(t)) \tag{2.7}$$

Ada dua metode untuk menyelesaikan persamaan kinemtika yaitu dengan metode aljabar dan geometris, pendekatan aljabar untuk memecahkan pada dasarnya memanipulasi persamaan yang diberikan ke dalam bentuk yang solusinya telah diketahui. Pendekatan geometris, kita menguraikan geometri masing – masing lengan kedalam beberapa bidang geometri[6]. Untuk lebih memudahkan untuk bisa menyelesaikan persamaan kinematik dengan geometri digunakan transformasi matriks atau yang biasa disebut transformasi homogenus.

2.3.1 Tranformasi Homogenus

Sebuah konfigurasi robot akan menghasilkan gerakan yang berbeda dengan konfigurasi lain, sehingga diperlukan cara untuk menganalisa gerakan dari robot tersebut. Untuk menganalisa gerakan robot biasanya digunakan pendekatan transformasi geometri pada sistem koordinat. Transformasi ini disesuaikan dengan perilaku masing – masing bagian mekanik penyusun robot terutama *joint* robot. Bagian ini adalah bagian yang bergerak dan memiliki dua macam gerakan yaitu memutar (rotasi) dan bergeser (translasi), sehingga untuk merepresentasikan gerakan pada *joint* tersebut, maka pada sumbu koordinat yang terlibat dalam sistem koordinat robot juga akan diperlakukan seperti *joint* yaitu diputar dan digeser sesuai dengan jenis *joint* yang digunakan pada konfigurasi robot yang terbentuk.

Pada robot ketika masing – masing *joint* bergerak maka *end - effector* juga akan ikut bergerak. Pergeseran *end - effector* berpengaruh pada posisinya, sedangkan gerakan memutar berpengaruh pada orientasinya. Untuk memudahkan analisa, maka diperlukan suatu bentuk transformasi yang bisa memetakan posisi dan orientasi dari *end – effector*. Transformasi yang bisa memetakan posisi dan orientasi secara bersamaan adalah transformasi homogen yang merupakan sebuah matriks yang berdimensi 4 x 4 yang tersusun atas 4 submatriks, seperti pada persamaan (2.8)

$$H = \begin{bmatrix} R_{3x3} & d_{3x1} \\ f_{1x3} & s_{1x1} \end{bmatrix} = \begin{bmatrix} Rotation & Translation \\ Perspective & Scale\ Factor \end{bmatrix} \quad (2.8)$$

Matriks transformasi homogen rotasi terhadap sumbu x, y dan z dapat dilihat pada persamaan (2.9) – (2.11) sedangkan untuk translasi pada sumbu x, y dan z dapat dilihat pada persamaan (2.12) – (2.14).

$$Rot_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

$$Rot_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.10)$$

$$Rot_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.11)$$

$$Trans_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.12)$$

$$Trans_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.13)$$

$$Trans_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.14)$$

2.3.2 *Forward* Kinematik

Forward kinematik adalah suatu metode yang digunakan untuk analisa gerak jika input yang di ketahui adalah sudut dari setiap variabel *joint* robot kemudian diingikan mencari nilai dari posisi dan orientasi dari *end – effector*. Dengan menggunakan metode kinematika pemodelan dari suatu gerakan menjadi lebih mudah untuk diselesaikan. Untuk mencari *forward* kinemarik ada beberapa tahapan dengan aturan umum yaitu *Denavit – Hartenberg* (DH) parameter. Aturan ini memberikan langkah – langkah yang mudah untuk mencari nilai kinematika maju. Pada aturan ini menyatakan bahwa tiap matriks transformasi homogen dinyatakan sebagai perkalian dari empat transformasi dasar yang melibatkan link dan joint robot, selain itu aturan ini juga mengetur mengenai penetapan kerangka koordinat robot. Aturan DH parameter dapat dijelaskan sebagai berikut :

a_i = jarak sepanjang x_i dari o_i ke perpotongan sumbu x_i dan z_{i-1}.

d_i = jarak sepanjang z_{i-1} dari o_{i-1} ke perpotongan sumbu x_i dan z_{i-1}. d_i berupa variabel jika *joint*i adalah *prismatic joint*.

α_i = sudut antara z_{i-1} dan z_i diukur terhadap x_i.

θ_i = sudut antara x_{i-1} dan x_i diukur terhadap z_i, θ_i berupa variabel jika *joint*$_i$ adalah *revolute joint*.

Langkah – langkah untuk mendapatkan *forward* kinematik menggunakan aturan DH parametes adalah sebagi berikut :

1. Tempatkan dan beri label z_0,. . . z_{n-1} pada sumbu putar *joint* atau sumbu geser *joint*.
2. Tetapkan Base Frame. Tentukan titik *origin* pada sumbu z_0. Sumbu z_0. Sumbu x_0 dan y_0 dipilih sembarang sehingga membentuk *right-hand frame*.
3. Tempatkan *origino* $_i$ ke z_i dan z_{i-1} memotong z_i.
4. Tetapkan x_i sepanjang *common normal* antara z_{i-1} dan z_i melalui o_i.
5. Tetapkan y_i untuk melengkapi *right-hand frame*.
6. Tetapkan *end-effector frame* pada o_n, x_n, y_n, z_n.
7. Buat Tabel parameter *link* a_i, d_i, α_i, θ_i.
8. Bentuk matriks transformasi homogen A_n dengan melakukan substitusi parameter.
9. Bentuk matriks *forward* kinematik $T_0^n = A_1 \ . \ . \ . \ A_n$. Matriks ini memberikan keluarab berupa posisi dan orientasi dari *end – effector frame* dalam koordinat dasar.

Berdasarkan aturan DH parameter, setiap transformasi homogen direpresentasikan sebagai hasil dari empat transformasi dasar seperti yang ditunjukkan pada Persamaan (2.15).

$$ {}^{i-1}{}_{i}T = Rot_{z,\theta i}.\, Trans_{z,di}.\, Trans_{x,ai}.\, Rot_{x,\alpha i} $$

$$ = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

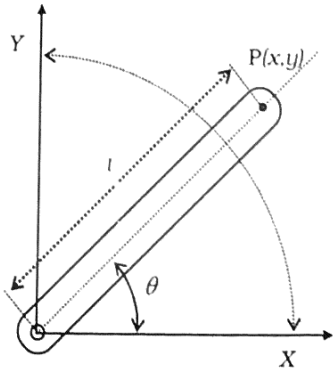
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}c_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} R_i & p_i \\ 0 & 1 \end{bmatrix} \quad (2.15)$$

2.3.3 *Invers* Kinematik

Invers kinematik adalah metode analisa geometri kebalikan dari *forward* kinematik input referensi dari *invers* kinematik berupa posisi yang merepresentasikan posisi dari *end – effector* robot, kemudian variable besarannya nilai sudut joint adalah nilai yang akan didapatkan pada analisa ini. Dalam mencari penyelesaian invers kinematik memiliki banyak sekali solusi dan bisa tidak memiliki solusi penyelesaiannya. Metode yang sering digunakan adalah metode dengan menggunakan analisa geometri yang biasa disebut kinematik *decoupling*. Pada metode ini analisanya yaitu dengan memecah masalah menjadi dua bagian yaitu *invers* posisi dan *invers* orientasi, *invers* posisi dapat diselesaikan dengan analisa geometri terutama trigonometri, sedangakan *invers* orientasi diselesaikan dengan pendekatan sudut Euler pada matriks rotasi. Gambar (2.3) merupakan contoh dari penyelesaian invers kinematik menggunakan analisa geometri.



Gambar 2.15 *Invers* Kinematik Pendekatakan Geometri

Pada gambar (2.15) ditetapkan posisi koordinat sumbu x dam y yang nantinya akan di cari berapa sudut θ menggunakan analisa geometri trigonometri pada persamaan (2.16 – 2.18).

$$x = l.\cos(\theta) \tag{2.16}$$

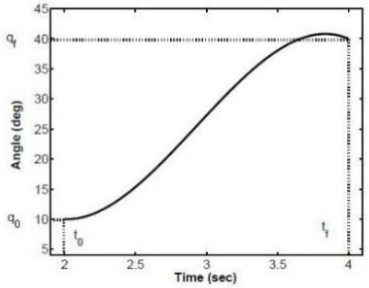
$$y = l.\sin(\theta) \tag{2.17}$$

Jika (x,y) diketahui maka θ dapat dihitung sebag)ai berikut,

$$\theta = arctan\frac{y}{x} \tag{2.18}$$

2.3.4 *Trayectori Planning*

Trayectori Planning adalah perencanan jalur gerak robot yang terdiri dari kumpulan titik – titik yang akan di lalui oleh end – effector robot ketika bergerak. Dengan membuat trayektori gerak robot menjadi lebih terarah dan lebih halus saat bergerak. Trayektori mendeskripsikan posisi ujung kaki sebagai fungsi waktu dalam sistem koordinat kaki. Salah satu cara untuk membuat jalur gerak robot yang halus adalah dengan menggunakan fungsi polinomial dengan empat koefisien yang dapat di tentukan yang di sebut dengan *trajectory cubic* gambar (2.16).



Gambar 2.16 *Trajectory cobic* pada joint robot

Bentuk polinomial pada trajectory cubic dapat dilihat pada persamaan (2.16)

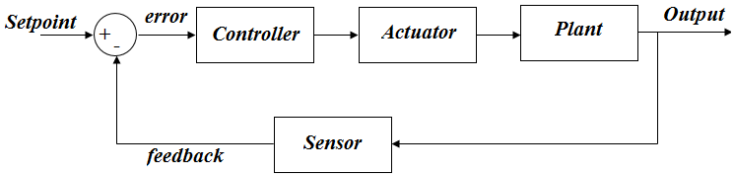
$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (2.19)$$

Persamaan (2.19) merupakan persamaan polinomial *trajectory* posisi yang merepresentasikan titik – titik jalur bergerak pada *end - effector* pada tiap satu siklus berjalan (*transefer phase* dan *support phase*). Dengan menurunkan persamaan (2.19) dapat diperoleh polinomial *trajectory* kecepatan yang dapat dilihat pada persamaan (2.20)

$$q'(t) = a_1 t + 2a_2 t + 3a_3 t^2 \quad (2.20)$$

2.4 Sistem Pengaturan

Sistem adalah kumpulan dari elemen yang saling berhubungan dan bekerja sama untuk menghasilkan tujuan tertentu. Sistem sendiri sangatlah luas tidak hanya pada suatu aspek tertentu saja. Pengaturan adalah suatu tindakan yang dilakukan supaya kondisi – kondisi tertentu dapat sesuai dengan keadaan yang diharapakan atau menjaga kondisi agar tetap pada tingkat yang kita inginkan.sehingga sistem pengaturan adalah suatu kumpulan dari elemen – elemen yang saling berhubungan dan bekerja sama untuk dapat menjaga kondisi – kondisi agar tetap pada tingkatan yang sesuai dengan apa yang diharapkan. Gambar (2.17) adalah salah satu dari gambaran sistem pengaturan



Gambar 2.17 Diagram Blok Sistem Pengaturan

2.4.1 Bagian Sistem Pengaturan

Seperti yang telah dijelaskan pada poin 2.4 bahwa sistem pengaturan adalah sekumpulan elemen – elemen yang saling berhubungan dan saling bekerja sama untuk dapat menjga suatu kondisi agar tetap pada tingkatan yang sesuai. Elemen – elemn dari sistem pengaturan ada beberapa elemen yang saling terhubung dapat di jelaskan sebagai berikut:

a. *Input* atau *Set Point*
   Input adalah nilai yang ditetapkan atau nilai yang diinginkan agar kondisi output dari sistem tersebut berada pada tingkatan nilai yang diinginkan atau *Input = Ouput*
b. *Error* detektor
   Error detektor merupakan komponen yang digunakan untuk membandingkan nilai *input* atau *setpoint* dengan nilai *output* yang terukur.
c. Kontroler
   Kontroler adalah komponen yang berfungsi sebagai pengolah sinyal agar *error* yang terbaca bernilai nol sehingga tidak ada selisih antara *input* dan *output* yang terukur
d. Aktuator
   Aktuator adalah komponen penguat dan pengkonversi daya yang berfungsi untuk menguatkan sinyal kontrol yang berasal dari kontroler menjadi sinyal yang dapat digunakan oleh *plant*
e. *Plant*
   Plant adalah suatu perangkat yang bekerja bersama yang digunakan untuk melakukan suatu operasi tertentu. Pada sistem pengaturan objek yang dikontrol adalah *plant*
f. Proses

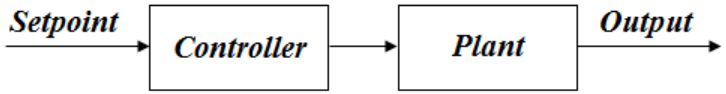
Proses adalah operasi pengembangan yang bgerlangsung secara kontinyu yang ditandai dengan deretan perubahan kecil yang berurutan dengan cara relatif tetap dan menuju suatu hasil atau keadaan tertentu. Pada umnya setiap operasi yang dikontrol disebut proses

g. *Feedback* atau Elemen Ukur
Feedback biasa disebut dengan umpan balik merupakan perangkat fisik yang mampu mengkonversi energi ke energi yang lainnya yang digunakan untuk membaca atau mengukur nilai output yang nantinya akan dibandingkan dengan nilai *input*.

h. *Output*
Output adalah suatu nilai yang dikeluarkan oleh sistem. Nilai ini digunakan untuk membandingkannya dengan nilai *input* sehingga didapatkan nilai errornya.

2.4.2 Sistem Pengaturan *Loop* Terbuka

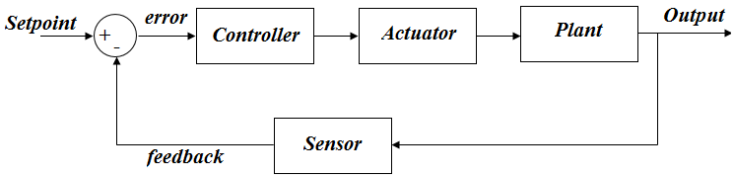
Sistem pengaturan loop terbuka merupakan suatu sistem pengaturan yang keluarannya tidak mempunyai pengaruh terhadap aksi kontrol. Pada sistem pengaturan loop terbuka keluarannya tidak dapat digunakan sebagai perbandingan umpan balik dengan masukan acuan (set point). Diagram blok sistem pengaturan loop terbuka ditunjukkan oleh gambar (2.18).



Gambar 2.18 Diagram Blok Sistem Pengaturan *Loop* Terbuka

2.4.3 Sistem Pengaturan *Loop* Tertutup

Sistem pengaturan loop tertutup merupakan suatu sistem pengaturan dimana sinyal keluaran mempunyai pengaruh langsung terhadap aksi kontrol. Pada sistem pengaturan loop tertutup terdapat jaringan umpan balik dari sinyal keluaran yang akan dibandingkan dengan sinyal masukan acuan sehingga menghasilkan sinyal error yang akan menjadi masukan dari kontroler. Diagram blok sistem pengaturan loop tertutup ditunjukkan oleh Gambar (2.19).



Gambar 2.19 Diagram Blok *Loop* Tertutup

2.5 Fuzzy Logic

Logika *fuzzy* adalah salah satu bentuk pendekatan dimana representasi suatu kejadian didistribusikan kedalam sejumlah istilah bahasa (yang dinyatakan dalam level kualitatif) dengan nilai kebenaran yang terletak antara 0 sampai 1. Teori sistem *fuzzy* dikembangkan oleh Lotfi Zadeh dari *University of California – Barkeley* pada tahun 1965. Konsep logika *fuzzy* menggantikan konsep "benar-salah" dari logika *boolean* menjadi derajat tingkat kebenaran. Oleh karena itu, konsep logika *fuzzy* sesuai dengan pola pikir manusia yang cenderung menilai suatu objek secara samar.

2.5.1 Himpunan *Fuzzy*

Himpunan *fuzzy* adalah rentang nilai antara salah dan benar untuk setiap anggotanya atau suatu himpunan yang beranggotakan sejumlah istilah dalam pengertian bahasa yang menyatakan level kualitatatif dari semesta pembicaraan. Misalkan pengukuran kecepatan motor dapat didistribusikan kedalam beberapa istilah bahasa yang menyatakan level kualitatif dari kecepatan motor, maka semesta pembicaraan untuk kecepatan motor dapat dibuat dalam bentuk suatu himpunan *fuzzy* yaitu "Sangat Lambat", "Lambat", "Sedang", "Cepat" dan "Sangat Cepat".

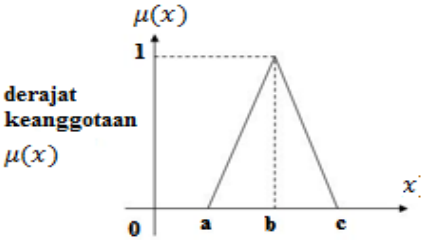
2.5.2 Fungsi Keanggotaan *Fuzzy*

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik – titik *input* ke dalam tingkat keanggotaan dengan nilai antara 0-1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai kenaggotaan adalah dengan melalu pendekatan fungsi [7]. Fungsi keanggotaan yang biasa

digunakan untuk merepresentasikan nilai keanggotan dari suatu himpunan *fuzzy* yaitu representasi kurva segtiga, representasi kurva trapesium, representasi kurva Gauss, representasi kurva-S dan representasi kurva lainnya.

- Fungsi Keanggotaan Segitiga
  Fungsi keanggotaan segitiga ditentukan oleh tiga parameter $\{a, b, c\}$ seperti yang ditunjukkan pada persamaan (2.21) sedangkan fungsi keanggotaan segitiga terlihat pada Gambar 2.20.

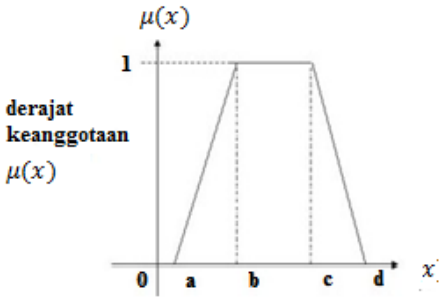
$$\mu(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & x > c \end{cases} \quad (2.21)$$



Gambar 2.20 Fungsi Keanggotaan Kurva Segitiga

- Fungsi Keanggotaan Trapesium
  Berbeda dengan fungsi keanggotaan segitiga yang ditentukan oleh tiga parameter, fungsi keanggotaan trapesium ditentukan oleh 4 parameter yaitu $\{a, b, c, d\}$ seperti yang pada persamaan (2.22) dan fungsi keanggotaan trapesium terlihat pada Gambar 2.21.

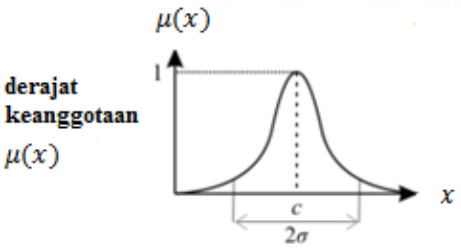
$$\mu(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \frac{c-x}{c-b}, & c \le x \le d \\ 0, & x > d \end{cases} \quad (2.22)$$



Gambar 2.21 Fungsi Keanggotaan Kurva Trapesium

- Fungsi Keanggotaan *Gaussian*
  Fungsi Keanggotaan *Gaussian* bergantung pada dua parameter yaitu$\{\sigma, c\}$. Parameter σ(standar deviasi) mendefinisikan lebar fungsi keanggotaan dan c menentukan pusat fungsi kenggotaan. Derajat keanggotaannya ditentukan sebagai berikut :

$$\mu(x) = e^{-\left(\frac{x-c}{\sigma}\right)^2} \quad (2.23)$$



Gambar 2.22 Fungsi Keanggotaan Kurva Gauss

2.5.3 Operasi Himpunan *Fuzzy*

Operasi himpunan *fuzzy* dilakukan untuk mengoperasikan fungsi keanggotaan yang satu dengan yang lain. Terdapat beberapa operator *fuzzy* yaitu operasi (*AND*), *union* (*OR*), komplemen (*NOT*), dan operasi lainnya. Misalkan terdapat dua himpunan *fuzzy*, yaitu A dan B dengan fungsi keanggotaan μ_A dan μ_B maka operasi himpunan *fuzzy* didefinisikan sebagai berikut:

1. Operasi (*AND*)

Operator ini berhubungan dengan operasi interseksi pada himupnan. Hasil operasi dengan operator AND diberikan operasi *minimum* dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan – himpunan yang bersangkutan [7]. Jika menggunakan operator *minimum* maka hasil operasi dapat dinyatakan sebagai berikut :

$$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}\,, x \in X \tag{2.24}$$

2. *Union* (*OR*)

Operator ini berhubungan dengan operasi *union* pada himpunan. Hasil operasi dengan operator OR diperoleh dengan mengambil nilai keanggotaan terbesar antar elemen pada himpunan – himpunan yang bersangkutan [7]. Jika operator yang digunakan adalah operator *maximum*, maka fungsi keanggotaannya sebagai berikut :

$$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B\,(x)\}\,, x \in X \tag{2.25}$$

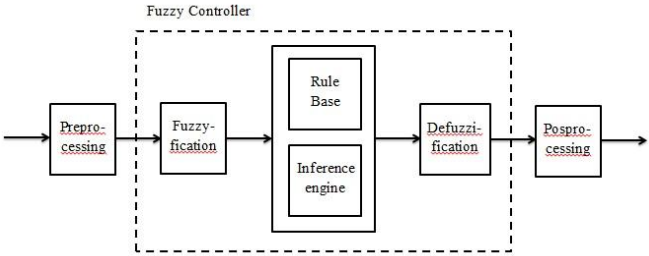
3. Komplemen (*NOT*)

Operasi komplemen pada sebuah himpunan *fuzzy* adalah hasil dari 1 dikurangi dengan fungsi keanggotaan dari himpunan *fuzzy*.

$$\mu'_A = 1 - \mu_A(x), x \in X \tag{2.26}$$

2.5.4 Kontroler Logika *Fuzzy*

Kontroler logika *fuzzy* merupakan suatu kontroler yang proses perhitungan sinyal kontrolnya dilakukan melalui operasi himpunan *fuzzy* meliputi proses *fuzzifikasi*, operasi dan relasi *fuzzy*, inferensi *fuzzy* serta *defuzzifikasi* seperti terlihat pada Gambar 2.23.

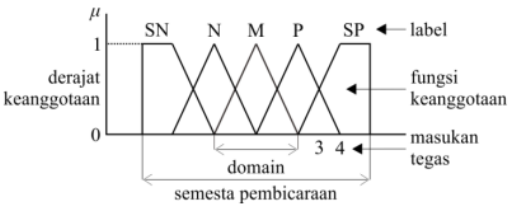


Gambar 2.23 Proses Kontroler Logika *Fuzzy*

Proses di dalam himpunan *fuzzy* pada kontroler *fuzzy* antara lain sebagai berikut :

1. *Fuzzifikasi*

Proses penguahan nilai *input* menjadi nilai *fuzzy* yang memiliki nilai antara 0 s/d 1. Aktivasi nilai *fuzzy* dilakukan dengan menggunakan fungsi keanggotaan *fuzzy*. Struktur fungsi keanggotaan *fuzzy* dapat dilihat pada Gambar 2.24. Fungsi keanggotaan mendefinisikan nilai *fuzzy* dengan melakukan pemetaan nilai tegas berdasarkan daerahnya untuk diasosiasikan dengan derajat keanggotaan. Jumlah fungsi keanggotaan *fuzzy* akan berpengaruh pada *output* kontroler *fuzzy*.



Gambar 2.24 Struktur Fungsi Keanggotaan *Fuzzy*

2. *Fuzzy Rule Base*

Basis aturan (*rule base*) merupakan deskripsi linguistik terhadap variabel *input* dan *output*. Pemetaan *input* dan *output* pada sistem *fuzzy* direpresentasikan dalam ***If*** *Premise* ***Then*** *Consequent*. Jumlah basis aturan dari suatu sistem *fuzzy* ditentukan dari jumlah variabel

pada input dan jumlah *membership function* pada variabel masukkan yang dirumuskan dalam Persamaan 2.27.

$$\Pi_{i=1}^{n} N_i = N_1 x N_2 x N_3 x \ldots x N_n \quad (2.27)$$

Dimana N_i merupakan jumlah *membership function* pada variabel *input* i. Sebagai contoh apabila variabel *input* pertama memiliki tiga *membership function* dan variabel *input* kedua memiliki tiga *membership function*, maka jumlah basis aturan adalah 3x3 = 9 aturan.

3. *Fuzzy Inference*

Terdapat beberapa tipe mekanisme inferensi *fuzzy* antara lain Mamdani, Larsent dan Takagi sugeno. Perbedaan dari metode ini terletak pada pengambilan kesimpulan logika *fuzzy*. Pada metode Mamdani maupun Larsent, kesimpulan logika *fuzzy* berupa derajat keanggotaan sehingga dalam menyimpulkan suatu logika *fuzzy* dibutuhkan proses *defuzzifikasi* sedangkan pada Takagi Sugeno, kesimpulan logika *fuzzy* berupa suatu persamaan sehingga tidak diperlukan proses *defuzzifikasi*.

4. *Defuzzifikasi*

Defuzzifikasi merupakan suatu proses mentransformasikan harga *fuzzy* hasil dari inferensi *fuzzy* ke dalam harga bukan *fuzzy* atau harga aktual. Beberapa metode dalam proses *defuzzifikasi* yaitu *Center of Area, Mean of Maxima,* dan lainnya

- *Center of Area*

Metode *center of area* digunakan untuk menentukan nilai titik tengah area yang merupakan titik pusat massa dari kombinasi fungsi-fungsi keanggotaan. Secara umum, persamaan untuk metode *Center of Area* ditunjukkan dengan Persamaan 2.28.

$$U_0 = \frac{\sum_{k=1}^{m} u_k(T) \bullet \mu_u(u_k(T))}{\sum_{k=1}^{m} \mu_k(u_k(T))}, \forall\ u \in U(T) \quad (2.28)$$

- *Mean of Maxima*

Metode *mean of maxima* mengambil semua nilai tiap fungsi keanggotaan dengan derajat keanggotaan maksimum dan menghitung rata-rata dari nilai-nilai tersebut sebagai keluaran tegas. Persamaan 2.29 menunjukkan persamaan umum metode tersebut.

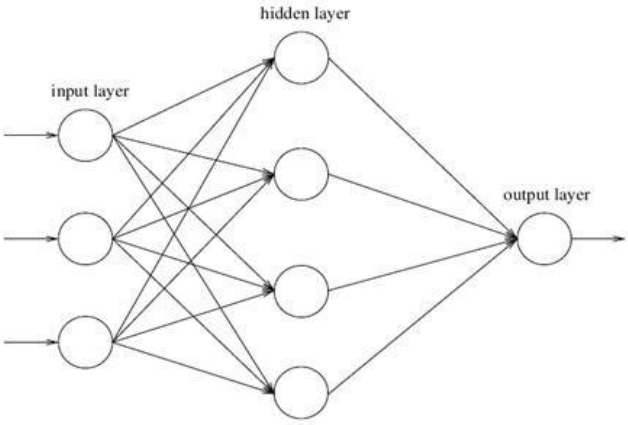
$$Uo = \frac{\sum_{n} max(\mu_A^n).y_n}{\sum_{n} max(\mu_A^n)} \quad (2.29)$$

2.6 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran [7]. Proses komputasi pada jaringan syaraf tiruan ini dipelajari dari struktur dan cara kerja otak manusia. Pada jaringan syaraf otak manusia, informasi disalurkan dari satu *neuron* ke *neuron* lainnya. Sementara pada jaringan syaraf tiruan proses penyaluran informasi dari satu *neuron* ke *neuron* lainnya diimplementasikan pada program komputer. Proses pelatihan jaringan syaraf tiruan, umumnya menggunakan metode pelatihan *backpropagation* yang sudah banyak diterapkan pada semua proses pelatihan yang sederhana sampai yang rumit.

Metode pelatihan *backpropagation* termasuk ke dalam metode pelatihan terawasi (*supervisory learning*). *Supervisory learning* adalah metode pelatihan yang memasukan target ke*output* dalam data untuk proses pelatihannya. Metode *backpropagation* banyak diaplikasikan dalam berbagai proses karena metode ini didasarkan pada interkoneksi yang sederhana. Apabila *output* jaringan syaraf tiruan tidak sesuai dengan ke*output* yang diinginkan, maka metode *backpropagation* akan memperbaiki niai bobot (*weight*) yang ada pada lapisan tersembunyi (*hidden layer*) untuk mencapai ke *output* jaringan syaraf tiruan yang sesuai dengan target ke *output*. Pada

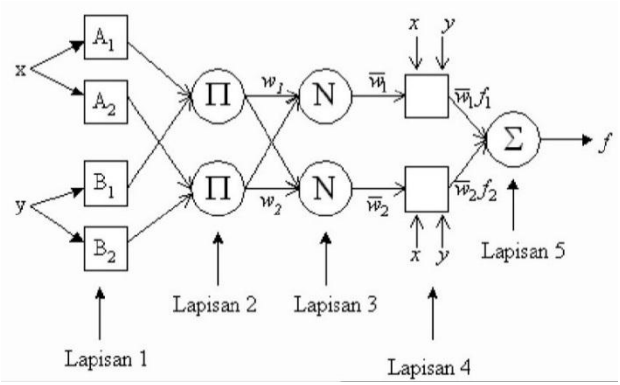
dasarnya jaringan syaraf tiruan akan diberikan pola masukan sebagai pola pelatihan maka pola akan menuju ke unit-unit lapisan tersembunyi (*hidden layer*) dan akan diteruskan ke lapisan *output*. Struktur jaringan syaraf tiruan dapat dilihat pada Gambar 2.25.



Gambar 2.25 Struktur *Neural Network*

2.7 Kontroler *Neuro – Fuzzy*

Neuro – Fuzzy adalah sistem yang menggabungkan logika *fuzzy logic* dengan jaringan syaraf tiruan, dimana nilai masukan dari jaringan syaraf diolah lebih dahulu melalui modul *fuzzifier* yang membuat nilai angka biasa menjadi nilai samar. Operasi dalam jaringan syaraf semua dalam nilai samar, kemudian keluarannya dikembalikan ke nilai biasa melalui modul *defuzzifier*. Metode *neuro – fuzzy* pada penelitian ini digunakan untuk mengontrol kestabilan tubuh robot hexapod dalam kondisi diam maupun berjalan pada medan yang tidak rata, bergelombang dan bidang miring.



Gambar 2.26 Struktur *Neuro - Fuzzy*

Lapis ke-1 : Setiap simpul pada lapisan ini adalah simpul adaptif dengan fungsi simpul

$O1,i = \mu Ai\ (x)$ untuk i = 1,2 (2.30)

$O1,i = \mu Bi\ (x)$ untuk i = 3,4 (2.31)

Fungsi O1i berfungsi untuk menyatakan derajat keanggotaan tiap masukan terhadap himpunan *fuzzy* A dan B[4].

Lapis ke-2 : Setiap simpul pada lapisan ini adalah simpul tetap yang merupakan perkalian dari sinyal yang datang. Operator perkalian dari aturan fuzzy pada simpul ini adalah AND[4].

$O2,i = Wi = \mu Ai\ (x) \cdot \mu Bi\ (y)$ untuk i = 1,2 (2.32)

Lapis ke-3 : Setiap simpul pada lapisan ini adalah simpul tetap. Pada simpul ke-i menghitung ratio dari aturan derajat keanggotaan ke-i dengan jumlah dari aturan derajat keanggotaan, sehingga didapatkan rumus sebagai berikut : [4]

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2}, i = 1,2 \quad (2.33)$$

Lapis ke-4 : Setiap simpul pada lapisan ini adalah simpul adaptif dengan fungsi simpul adalah: [4]

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i) \quad (2.34)$$

Dimana : {pi, qi, ri} adalah himpunan parameter yang disebut *consequent parameter*

Lapis ke-5 : Merupakan simpul tunggal yang menghitung dengan cara menjumlahkan semua sinyal masukan[4].

$$O_{5,i} = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (2.35)$$

Back Propagation : Merupakan proses *learning* terhadap kesalahan atau *error* yang muncul, proses ini berfungsi untuk meng-*update* nilai *consequent* paramater pada lapisan 4 atau titik – titik puncak *fuzzifikasi* pada lapisan 1 atau titik – titik puncak *defuzzifikasi* pada lapisan 5, berdasarkan *learning rate* dan nilai *error* yang terjadi. Proses pembelajaran ini bisa di lakukan pada semua lapisan atau pada salah satu lapisa saja.

fi = *consequent parameter* baru

f(i-1) = *consequent paramter* lama

lr = *learning rate*

e = *error* (*Reference* – output defuzzyfikasi)

Wi = *output* ternormalisasi lapisan 3

$$fi = f(i-1) + lr * e * Wi \quad (2.36)$$

2.8 Perangkat Pendukung

Robot *hexapod* memiliki peralatan pendukung yang penting untuk menggerakkan setiap sendi dari robot dibutuhkan motor servo yang mampu berpindah posisi sesuai dengan sudut yang kita masukkan, motor servo sendiri juga memerlukan sebuah driver untuk merubah atau mengkonversi sinyal kontrol dari kontroler sehingga dapat berubah sesuai dengan posisi sudut yang diberikan. Sebuah robot harus memiliki sebuah otak yang dapat menghitung dan memproses sinyal – sinyal untuk dapat berjalan dan lain sebagainya yang biasa disebut dengan kontroler perangkat pemroses sinyal atau

data. Pada penelitian kali ini diharapkan badan robot tetap stabil saat berjalan maupun diam pada kondisi *track* yang mirng dan tidak, oleh karena itu diperlukan sensor yang dapat membaca posisi kemiringan badan robot yang berupa nilai *roll, pitch* dan *yaw*, sehingga diperlukan sensor *IMU (Inertial Measuring Unit)* yang dapat menghasilkan nilai tersebut.

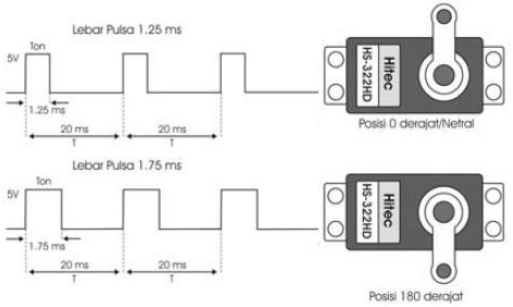
2.8.1 Servomekanisme Robot Hexapod

Servomekanisme yang terintergrasi sering disebut dengan servo, salah satu jenis aktuator yang digunakan pada penelitian ini yaitu menggunakan motor servo posisi. Servo ini adalah sebuah motor DC yang di desain khusus untuk aplikasi pengedalian posisi sudut dengan kontrol yang sudah terintegrasi di dalamnya, selain itu komponen dari motor servo ini sendiri adalah potensiometer sebagai pembaca atau sensor posisi sudut untuk sebagai umpan balik yang akan di baca sinyal outputnya oleh kontroler didalamnya sehingga mampu bergerak sesuai dengan posisi yang di berikan. Servo ini di rancang sampai dengan 50 Kg/cm torsi maksimal sesuai dengan jenis servo itu sendiri. Torsi ini diperoleh dengan mereduksi kecepatan putar poros motor DC menggunakan *gearbox* dengan tingkat perbandingan yang tinggi. Gambar 2.27 merupakan konstruksi motor servo.

Gambar 2.27 Konstruksi Motor Servo

Pada umumnya servo hanya memiliki 3 kabel dengan spesifikasi untuk VCC, GND dan sinyal PWM pengendali. Prinsip kerja dari motor servo ini adalah berdasarkan sinyal PWM yang di berikan dengan *duty cycle* tertentu maka akan mengeluarkan output sudut dengan nilai tertentu. Lebar pulsa duty cycle dan potensiometer inilah yang menyebabkan motor DC dapat berputar dan berhenti tepat pada posisi sudut tertentu meskipun perlu dilakukan kalibrasi terlebih

dahulu untuk lebih meningkatkan tingkat akurasi motor servo. Bisa dilihat pada gambar 2.28.



Gambar 2.28 Sinyal Kontrol (PWM) Motor Servo

2.8.2 *Driver* Servo

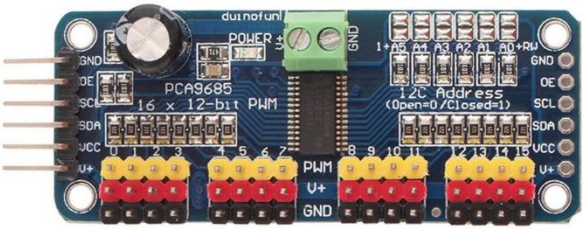
Driver motor servo ini menggunakan IC PCA9865 yang menggunakan komunikasi i2c sebagai media untuk transfer data dari kontroler ke driver. Driver ini memiliki 16 channel PWM dengan resolusi 12 bits (4096). Untuk menggerakan motor servo menggunakan driver motor ini menggunakan nilai rentang frekuensi antara 40 – 1000 Hz. Input yang di berikan berupa nilai minimal dan maksimal yang di *mapping* berdasarkan nilai sudut minimal dan maksimal servo, contoh :

- pwm.setPWMFreq(1000) kode ini digunakan untuk menginisialisasi frekuensi yang akan di gunakan (1000 Hz)
- pwm.setPWM(15, 1024, 3072) kode ini menset nilai PWM yang akan di berika dengan resolusi sebesar 12 bits atau 4096. 15 merupakan *channel* output yang dipakai, 1024 merupakan nilai setting PWM *on* dan 3072 merupakan nilai setting PWM *off.*

Untuk membuat servo dapat bergerak 0 – 180 derajat maka harus dicari nilai PWM saat derajat 0 dan saat derajat 180 kemudian nilai tersebut kita *mapping,* contoh :

- *Pulse* = map(derajat, 0, 180, 1024, 4096) kode ini artinya yaitu nilai *pulse* pada saat 0 derajat sama dengan 1024 dan nilai pulse pada saat 180 derajat sama dengan 4096. Gambar

2.29 adalah bentuk *hardware* dari *driver* motor servo PCA9865



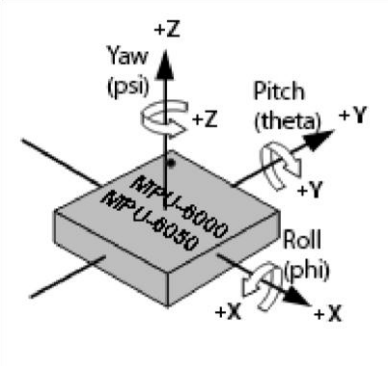
Gambar 2.29 Driver Motor Servo PCA9865

2.8.3 *Inertial Measuring Unit (IMU)*

Sebuah *Inertial Measurement Unit* yang selanjutnya disebut IMU merupakan sistem sensing benda yang digunakan untuk mengetahui perilaku benda dengan mengkombinasikan kerja sensor *accelerometer* tiga sumbu dan *gyroscope* tiga sumbu[7]. IMU yang akan digunakan pada tugas akhir ini adalah sensor MPU-6050 yang dikemas dalam modul GY521 dengan komunikasi data menggunakan *serial interfacing* I^2C. Dalam perancangan metode kontrol *quadcopter*, IMU merupakan faktor yang sangat penting karena dengan informasi sudut yang akurat koreksi terhadap kesalahan sistem akan lebih maksimal. Sampai pada saat ini, IMU masih menjadi topik yang hangat untuk diperbincangkan dan masih mengalami pengembangan yang samakin pesat. Selain untuk aplikasi *quadcopter*, IMU juga digunakan dalam banyak aplikasi mulai dari militer, satelit, *smart phone*, sampai aplikasi *gimbal* untuk pengambilan gambar maupun video.

Sensor *gyroscope* digunakan untuk mengindra (*sensing*) kecepatan putar sudut *roll* (*roll rate*), kecepatan putar sudut *pitch* (*pitch rate*) dan kecepatan putar sudut *yaw* (*yaw rate*). Kecepatan putaran adalah perubahan sudut terhadap waktu. *Accelerometer* digunakan untuk mengukur percepatan sebuah benda yang bergerak. Percepatan adalah perubahan kecepatan dalam suatu interval waktu. Percepatan pada *quadcopter* dihasilkan oleh gaya-gaya luar yang berkerja (*acting*) kepada *quadcopter*. Gaya luar ini terdiri dari gaya

dorong *propeller* (*thrust*), gaya aerodinamik (gaya angkat, *lift*, dan gaya friksi udara, *drag*.), dan gaya gravitasi bumi. Gaya dorong dan gaya aerodinamik ini membentuk gaya yang disebut gaya inersia. Gaya inersia dan gaya gravitasi bumi adalah gaya yang diukur oleh *accelerometer*. Gambar 2.30 menunjukkan orientasi *roll, pitch, yaw* pada sensor MPU-6050.

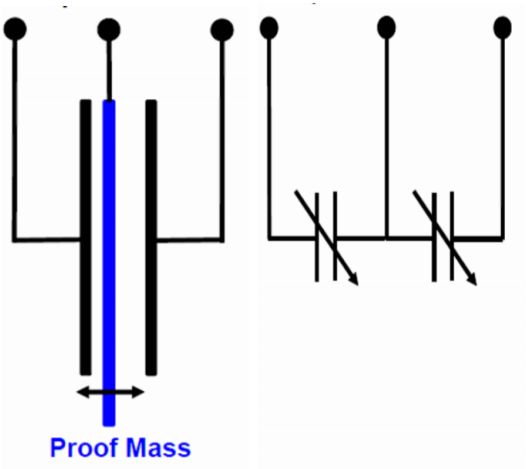


Gambar 2.30 Sensor MPU-6050 *Roll Pitch Yaw*

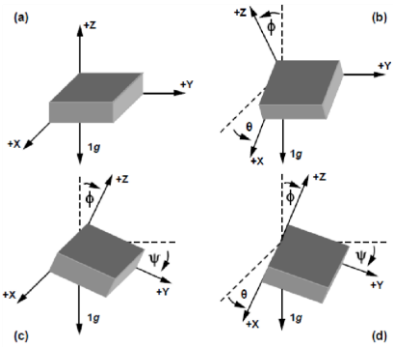
2.8.3.1 ***Accelerometer***

Sensor yang digunakan pada tugas akhir ini merupakan *accelerometer* digital yang mengukur percepatan pada 3 sumbu dengan satuan *g* (untuk gravitasi). Pada setiap *accelerometer*, datasheet akan memberikan keterangan mengenai vektor percepatan mengingat pembacaan sensor ini dipengaruhi juga oleh gaya gravitasi[7]. *Accelerometer Micro Electric Mechanic System* (MEMS) mempunyai model yang disederhanakan seperti yang ditunjukkan pada gambar 2.31. Dalam menentukan sudut yang dihasilakan sensor pada setiap sumbu, kemiringan harus ditentukan berdasarkan posisi referensi dengan permukaan horizontal pada sumbu-x dan sumbu-y (medan 0g) dan sumbu-z tegak lurus terhadap horizontal (medan 1g) merupakan sudut antara sumbu x akselerometer dengan horizontal, sebagai sudut antara sumbu y akselerometer dengan horizontal, dan merupakan sudut antara sumbu z akselerometer dengan horizontal dengan vektor grafitasi. Ketika posisi awal sumbu-x dan sumbu-y 0g dan posisi sumbu-z 1g, maka sudut yang terukur adalah 0˚. Pada

gambar 2.32 ditunjukkan sudut orientasi yang dihasilkan benda pada sumbu 3 dimensi dengan vector grafitasi pada sumbu z.



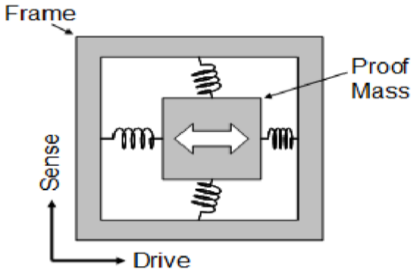
Gambar 2.31 Model *Accelerometer* yang Diserhanakan dan Rangkaian Ekuivalenya.



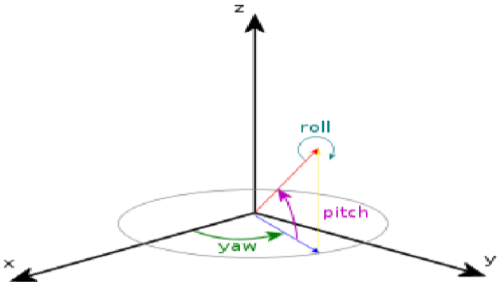
Gambar 2.32 Sudut Kemiringan *Accelerometer*

2.8.3.2 *Gyroscope*

Definisi dasar *gyroscope* menurut Matthew Eby dari University of Colorado adalah perangkat yang mengukur sudut orientasi baik itu seccara langsung maupun melalui integrasi dari pengukuran percepatan sudut. *Gyroscope* 3 *Degree of Freedom* (DOF) mengukur kecepatan sudut pada 3 sumbu xyz. Perbedaan nilai pada sensor kapasitif yang proporsional dengan kecepatan sudut kemudian dikonversi ke tegangan



Gambar 2.33 Model Sederhana *Gyroscope*



Gambar 2.34 *Roll Pitch Yaw*

untuk gyroscope analog atau LSBs untuk gyroscope digital. Gambar 2.33 digambarkan model sederhana sensor *gyroscope* pada perangkat *MEMS*.

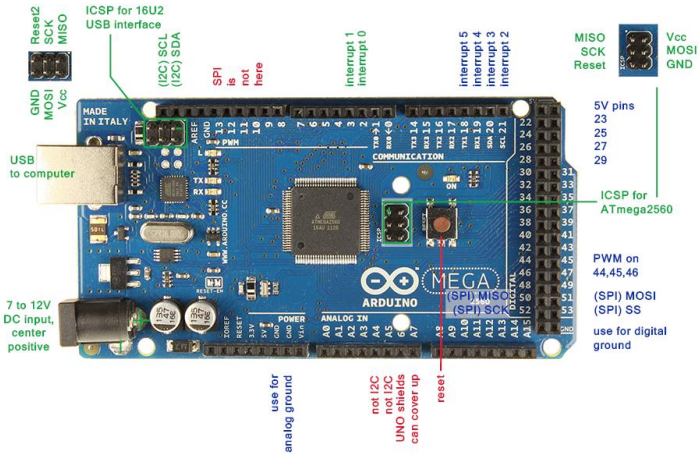
2.8.3.3 *Roll Pitch Yaw*

Perilaku benda pada ruang tiga dimensi bisa ditentukan dengan perubahannya berdasarkan posisi referensi. *Roll* merupakan rotasi yang terjadi pada sumbu-y, *pitch* merupakan rotasi pada sumbu-x,

sedangkan *yaw* merupakan rotasi pada sumbu-z. Nilai rotasi kecepatan sudut didapatkan dari sensor gyroscope yang memiliki satuan degree/second. Gambar 2.34 menunjukkan orientasi perubahan sudut *roll*, *pitch*, dan *yaw* pada kerangka sumbu tiga dimensi.

2.8.4 Arduino Mega 2560

Arduino Mega2560 adalah papan mikrokontroler berbasiskan ATmega2560). AT-Mega2560 memiliki 54 pin digital input/output, dimana 15 pin dapat digunakan sebagai *output* PWM, 16 pin sebagai *input analog*, dan 4 pin sebagai UART (*port serial hardware*), 16 MHz kristal osilator, koneksi USB, jack power, header ICSP, dan tombol reset. Ini semua yang diperlukan untuk mendukung mikrokontroler. Cukup dengan menghubungkannya ke komputer melalui kabel USB atau power dihubungkan dengan adaptor AC-DC atau baterai untuk mulai mengaktifkannya. Berikut ini adalah pemetaan pin Arduino Mega :



Gambar 2.35 Pemetaan Pin Arduino Mega 2560

2.8.4.1 Sumber Daya

Arduino Mega dapat diaktifkan melalui koneksi USB atau dengan catu daya eksternal. Sumber daya dipilih secara otomatis.

Sumber daya eksternal (non-USB) dapat berasal baik dari adaptor AC-DC atau baterai. Adaptor dapat dihubungkan dengan mencolokkan steker 2,1 mm yang bagian tengahnya terminal positif ke ke jack sumber tegangan pada papan. Jika tegangan berasal dari baterai dapat langsung dihubungkan melalui header pin Gnd dan pin Vin dari konektor *POWER*.

Papan Arduino ATmega2560 dapat beroperasi dengan pasokan daya eksternal 6 Volt sampai 20 volt. Jika diberi tegangan kurang dari 7 Volt, maka, pin 5 Volt mungkin akan menghasilkan tegangan kurang dari 5 Volt dan ini akan membuat papan menjadi tidak stabil. Jika sumber tegangan menggunakan lebih dari 12 Volt, *regulator* tegangan akan mengalami panas berlebihan dan bisa merusak papan. Rentang sumber tegangan yang dianjurkan adalah 7 Volt sampai 12 Volt.

Pin tegangan yang tersedia pada papan Arduino adalah sebagai berikut:

1. **VIN** : Adalah input tegangan untuk papan Arduino ketika menggunakan sumber daya eksternal (sebagai 'saingan' tegangan 5 Volt dari koneksi USB atau sumber daya ter-*regulator* lainnya). Anda dapat memberikan tegangan melalui pin ini, atau jika memasok tegangan untuk papan melalui jack power, kita bisa mengakses/mengambil tegangan melalui pin ini.
2. **5V** : Sebuah pin yang mengeluarkan tegangan ter-*regulator* 5 Volt, dari pin ini tegangan sudah diatur (ter-regulator) dari regulator yang tersedia (*built-in*) pada papan. Arduino dapat diaktifkan dengan sumber daya baik berasal dari jack power DC (7-12 Volt), konektor USB (5 Volt), atau pin VIN pada board (7-12 Volt). Memberikan tegangan melalui pin 5V atau 3.3V secara langsung tanpa melewati regulator dapat merusak papan Arduino.
3. **3V3** : Sebuah pin yang menghasilkan tegangan 3,3 Volt. Tegangan ini dihasilkan oleh regulator yang terdapat pada papan (on-board). Arus maksimum yang dihasilkan adalah 50 mA.
4. **GND** : Pin Ground atau Massa.
5. **IOREF** : Pin ini pada papan AT-Mega 2560 berfungsi untuk memberikan referensi tegangan yang beroperasi pada mikrokontroler. Sebuah perisai (*shield*) dikonfigurasi

dengan benar untuk dapat membaca pin tegangan IOREF dan memilih sumber daya yang tepat atau mengaktifkan penerjemah tegangan (voltage translator) pada output untuk bekerja pada tegangan 5 Volt atau 3,3 Volt.

2.8.4.2 Memori

ATmega2560 memiliki 256 KB flash memori untuk menyimpan kode (yang 8 KB digunakan untuk bootloader), 8 KB SRAM dan 4 KB EEPROM (yang dapat dibaca dan ditulis dengan perpustakaan EEPROM).

2.8.4.3 *Input* dan *Output*

Masing-masing dari 54 digital pin pada AT-Mega dapat digunakan sebagai input atau output, menggunakan fungsi pinMode(), *digitalWrite()*, dan *digitalRead()*. AT-Mega beroperasi pada tegangan 5 volt. Setiap pin dapat memberikan atau menerima arus maksimum 40 mA dan memiliki resistor *pull-up* internal (yang terputus secara *default*) sebesar 20-50 kOhms. Selain itu, beberapa pin memiliki fungsi khusus, antara lain:

1. **Serial** : 0 (RX) dan 1 (TX); **Serial 1** : 19 (RX) dan 18 (TX); **Serial 2** : 17 (RX) dan 16 (TX); **Serial 3** : 15 (RX) dan 14 (TX). Digunakan untuk menerima (RX) dan mengirimkan (TX) data serial TTL. Pins 0 dan 1 juga terhubung ke pin chip ATmega16U2 Serial USB-to-TTL.
2. **Eksternal Interupsi** : Pin 2 (interrupt 0), pin 3 (interrupt 1), pin 18 (interrupt 5), pin 19 (interrupt 4), pin 20 (interrupt 3), dan pin 21 (interrupt 2). Pin ini dapat dikonfigurasi untuk memicu sebuah interupsi pada nilai yang rendah, meningkat atau menurun, atau perubah nilai.
3. **SPI** : Pin 50 (MISO), pin 51 (MOSI), pin 52 (SCK), pin 53 (SS). Pin ini mendukung komunikasi SPI menggunakan perpustakaan SPI. Pin SPI juga terhubung dengan header ICSP, yang secara fisik kompatibel dengan Arduino Uno, Arduino Duemilanove dan Arduino Diecimila.
4. **LED** : Pin 13. Tersedia secara built-in pada papan Arduino ATmega2560. LED terhubung ke pin digital 13. Ketika pin diset bernilai HIGH, maka LED menyala (ON), dan ketika pin diset bernilai LOW, maka LED padam (OFF).

5. **TWI** : Pin 20 (SDA) dan pin 21 (SCL). Yang mendukung komunikasi TWI menggunakan perpustakaan Wire. Perhatikan bahwa pin ini tidak di lokasi yang sama dengan pin TWI pada Arduino Duemilanove atau Arduino Diecimila.

AT-Mega2560 memiliki 16 pin sebagai *analog input*, yang masing-masing menyediakan resolusi 10 bit (yaitu 1024 nilai yang berbeda). Secara *default* pin ini dapat diukur/diatur dari mulai *Ground* sampai dengan 5 Volt, juga memungkinkan untuk mengubah titik jangkauan tertinggi atau terendah mereka menggunakan pin AREF dan fungsi *analog reference*.
Ada beberapa pin lainnya yang tersedia, antara lain:

1. **AREF** : Referensi tegangan untuk input analog. Digunakan dengan fungsi *analogReference()*.
2. **RESET** : Jalur *LOW* ini digunakan untuk me-reset (menghidupkan ulang) mikrokontroler. Jalur ini biasanya digunakan untuk menambahkan tombol reset pada *shield*.

2.8.4.4 Perangkat Komunikasi

AT-Mega2560 memiliki sejumlah fasilitas untuk berkomunikasi dengan komputer, atau dengan mikrokontroler lainnya. ATmega328 menyediakan 4 hardware komunikasi serial UART TTL (5 Volt). Sebuah chip ATmega16U2 (ATmega8U2 pada papan Revisi 1 dan Revisi 2) yang terdapat pada papan digunakan sebagai media komunikasi serial melalui USB dan muncul sebagai COM Port Virtual (pada Device komputer) untuk berkomunikasi dengan perangkat lunak pada komputer, untuk sistem operasi Windows masih tetap memerlukan file inf, tetapi untuk sistem operasi OS X dan Linux akan mengenali papan sebagai port COM secara otomatis. Perangkat lunak mikrokontroler termasuk didalamnya serial monitor memungkinkan data tekstual sederhana dikirim ke dan dari mikokontroler board. LED RX dan TX yang tersedia pada papan akan berkedip ketika data sedang dikirim atau diterima melalui chip USB-to-serial yang terhubung melalui USB komputer (tetapi tidak untuk komunikasi serial seperti pada pin 0 dan 1).

Sebuah perpustakaan SoftwareSerial memungkinkan untuk komunikasi serial pada salah satu pin digital Mega2560. ATmega2560 juga mendukung komunikasi TWI dan SPI. Perangkat

lunak Arduino termasuk perpustakaan Wire digunakan untuk menyederhanakan penggunaan bus TWI. Untuk komunikasi SPI, menggunakan perpustakaan SPI.

2.8.4.5 Spesifikasi Arduino Mega2560

Berikut ini adalah spesifikasi dari Arduino Mega2560:

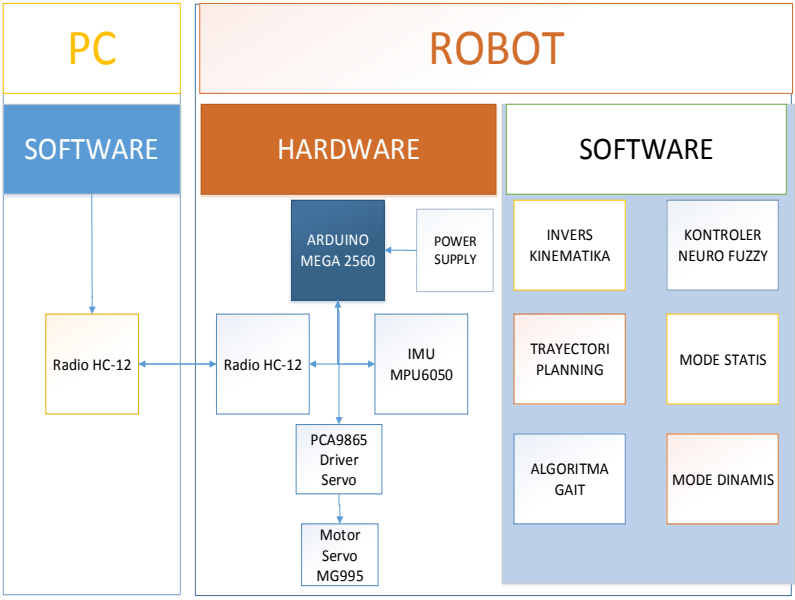
Tabel 2.1 Tabel Ringkasan Spesifikasi Arduino Mega 2560

Mikrokontroler	Atmega2650
Tegangan Operasi	5V
Input Voltage	7-12V
Input Volatge (limit)	6-2-V
Pin Digital I/O	54 (15 pin digunakan sebagai output PMW)
Pin Input Nlog	16
Arus DC per pin I/O	40Ma
Arus DC untuk pin 3.3V	50mA
Flash Memori	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

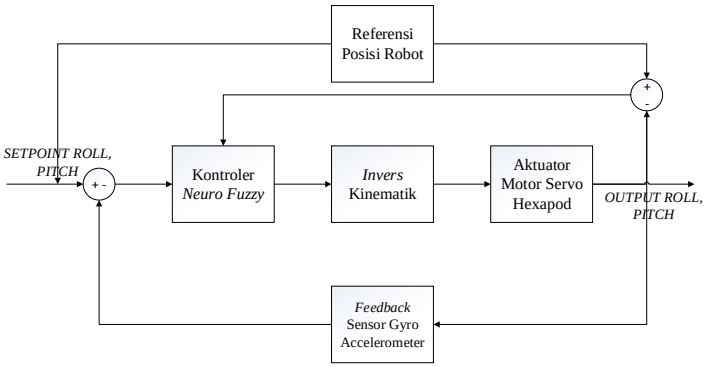
-- Halaman ini sengaja dikosongkan --

BAB III
PERANCANGAN SISTEM

Pada bagian ini akan dijelaskan bagaimana perancangan sistem yang akan dikerjakan pada penelitian ini yang terdiri dari konstruksi bentuk robot, paramter DH, model kinematika termasuk penyelesaiannya, perancangan kontroler keseimbangan, perancangan *hardware* seperti konfigurasi pin, *driver* motor servo dan *software* yang berisi algoritma kontrol keseimbangan, *gait*, *trayektori planning* dan semua yang berkaitan. Rancangan keseluruhan sistem dapat dilihat pada gambar 3.1 dan gambar 3.2 adalah blok diagram kontrol keseimbang hexapod.



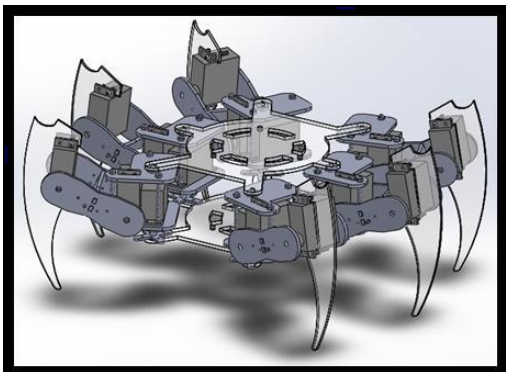
Gambar 3.1 Rancangan Sstem Pengaturan Keseimbangan Robot *Hexapod*



Gambar 3.2 Diagram Blok Pengaturan Keseimbangan Robot *Hexapod*

3.1 Konstruksi Robot *Hexapod*

Robot yang digunakan pada penelitian kali ini adalah robot hexapod *linxmotion* yang menggunakan bahan plastik akrilk. Dimensi saat berdiri: 280mm x 320mm x 175mm. Menggunakan 3 servo MG996R dan 15 servo MG995. Gambar 3.3 merupakan bentuk fisik dari robot hexapod *linxmotion*.

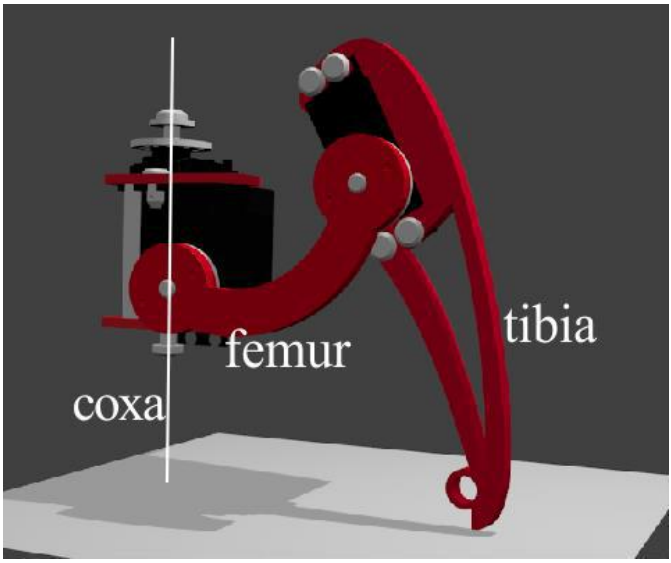


Gambar 3.3 Robot *Hexapod Linxmotion*

Robot *hexapod* adalah robot berkaki enam dengan 3 DOF *(Degree Of Freedom)* pada tiap masing - masing kaki nya. Untuk mempermudah pemodelan kinematikanya robot ini dibagi menjadi dua yaitu bagian kaki robot dan bagian badan robot.

3.2 Konstruksi Kaki Robot *Hexapod*

Semua joint pada robot ini adalah revolute joint yang artinya sudut putarnya berupa nilai variable yang memiliki nilai dengan batasan – batasan tertentu, setiap sendi atau joint robot menggunakan motor servo sebagai aktuator penggerak robot yang terdiri dari 3 motor servo pada setiap kaki robot *hexapod*, gambar 3.4 dibawah merupakan gambar kontruksi kaki robot *hexapod*.

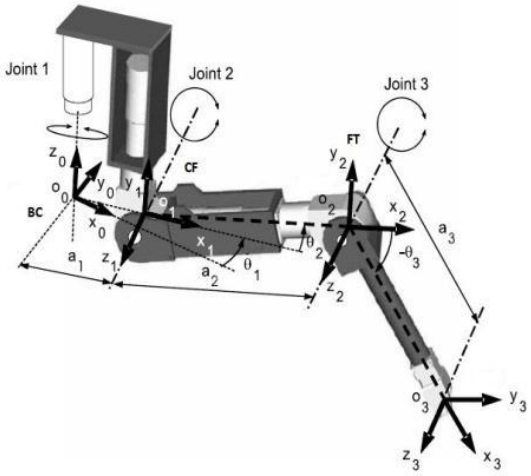


Gambar 3.4 Kontruksi Kaki Robot *Hexapod*

Pada gambar 3.3 robot *hexapod* memiliki 3 bagian kaki yang disederhanakan seperti serangga, yaitu *coxa*, *femur* dan *tibia*. Setelah didapatkan gambar konstruksi kaki robot maka hal selanjutnya dalam merancangan adalah membuat model kinematika dari konstruksi robot tersebut.

3.2.1 *Forward* Kinematik Kaki *Hexapod*

Forward kinematik merupakan suatu metode analisa kinematika yang digunakan untuk mencari posisi dari suatu robot manipulator dengan memberi *input* berupa nilai sudut – sudut pada setiap joint atau sendi robot. Salah satu metode yang sering digunakan untuk mencari penyelesaiannya adalah dengan menggunakan DH *(Denavid – Hartenberg)* parameter. Untuk memudahkan pemodelan kinematika robot, konstruksi kaki robot digambarkan seperti pada gambar 3.5.



Gambar 3.5 Representasi Konstruksi Kaki Robot *Hexapod*

Untuk menyelesaikan pemodelan kinematika ini diperlukan tabel parameter yang dapat dilihat pada tabel 3.1 Untuk mendapatkan nilai dari tabel 3.1 menggunakan langkah - langkah sebagai berikut:

1. Tempatkan dan beri label z_0,. . . z_{n-1} pada sumbu putar *joint* atau sumbu geser *joint*.
2. Tetapkan Base Frame. Tentukan titik *origin* pada sumbu z_0. Sumbu z_0. Sumbu x_0 dan y_0 dipilih sembarang sehingga membentuk *right-hand frame*.
3. Tempatkan *origino* $_i$ ke z_i dan z_{i-1} memotong z_i.

4. Tetapkan x_i sepanjang *common normal* antara z_{i-1} dan z_i melalui o_i.
5. Tetapkan y_i untuk melengkapi *right-hand frame*.
6. Tetapkan *end-effector frame* pada o_n, x_n, y_n, z_n.
7. Buat Tabel parameter *link* $a_i, d_i, \alpha_i, \theta_i$.
8. Bentuk matriks transformasi homogen A_n dengan melakukan substitusi parameter.
9. Bentuk matriks *forward* kinematik $T_0^n = A_1 \ . \ . \ . \ A_n$. Matriks ini memberikan keluarab berupa posisi dan orientasi dari *end – effector frame* dalam koordinat dasar.

Tabel 3.1 Parameter DH

LINK	Ai	αi	di	θi
1	25 mm (l_1)	90°	0	0 – 180
2	75 mm (l_2)	0°	0	0 – 180
3	121 mm (l_3)	0°	0	0 – 180

Nilai pada tabel 3.1 dihitung kedalam bentuk matrik transformasi Homogen Ai, untuk i = sejumlah link.

$$A_i = T_i^{i-1} \tag{3.1}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}c_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriks T adalah matriks transformasi T untuk *joint* 1 sampai dengan *joint* 3.

$$T_3{}^0 = A_1 A_2 A_3$$

$$T_3{}^0 = \begin{bmatrix} C_1 & 0 & S_1 & l_1C_1 \\ S_1 & 0 & -C_1 & l_1S_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & l_2C_2 \\ S_2 & C_2 & 0 & l_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & l_3C_3 \\ S_3 & C_3 & 0 & l_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

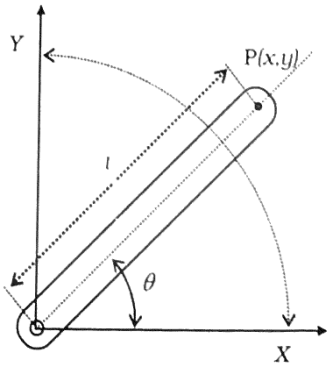
$$T_3{}^0 = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & C_1(l_1 + l_2C_2 + l_3C_{23}) \\ S_1S_{23} & -S_1S_{23} & -C_1 & S_1(l_1 + l_2C_2 + l_3C_{23}) \\ S_{23} & C_{23} & 1 & l_3S_{23} + l_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

Dari matriks T pada persamaan (3.2), maka persamaan *forward* kinematiknya pada persamaan (3.3).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} C_1(l_1 + l_2C_2 + l_3C_{23} \\ S_1(l_1 + l_2C_2 + l_3C_{23}) \\ l_3S_{23} + l_2S_2 \end{bmatrix} \quad (3.3)$$

3.2.2 *Invers* Kinemarik Kaki *Hexapod*

Invers kinematik robot *hexapod* dibagi menjadi 2 bagian untuk mempermudah orientasinya yaitu invers kinematik pada orientasi koordinat x dan y dan koordinat x dan z. Pada gambar 3.5 merupakan invers kinematik orientasi sumbu x dan y, gambar 3.6 invers kinematik orientasi sumbu x dan z.



Gambar 3.6 Representasi *Invers* Kinematik *Hexapod* Sumbu X, Y

Kedudukan ujung lengan P(x,y) dapat diperoleh dengan cara *forward* kinematik sebagai berikut,

$$x = l.\cos(\theta) \quad \textbf{(3.4)}$$

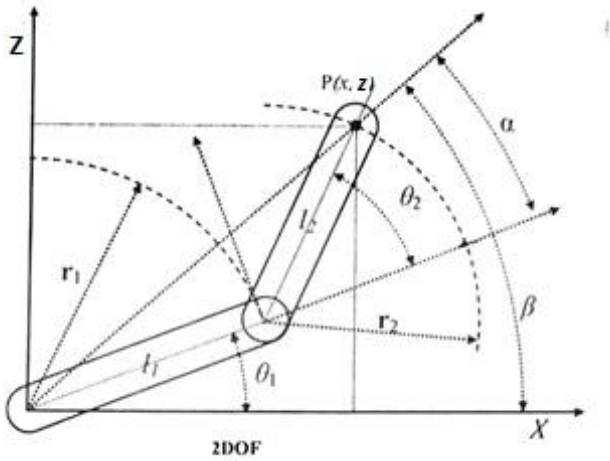
$$y = l.\sin(\theta) \quad \textbf{(3.5)}$$

Jika (x,y) diketahui maka θ dapat dihitung sebagai berikut,

$$\theta = arctan\frac{y}{x} \quad \textbf{(3.6)}$$

$$\theta = tan^{-1}\frac{y}{x}$$

Dari persamaam (3.6) didapatkan persamaan untuk mencari sudut $\theta 1$.



Gambar 3.7 Representasi *Invers* Kinematik *Hexapod* Sumbu X, Z

Jika P diasumsikan sebagai vektor penjumlahan yang terdiri dari vektor r1 lengan-1 dan r2 lengan-2,

$$r1 = [l_1\ cos\ \theta 1\ , l_1\ sin\ \theta 1] \quad \textbf{(3.7)}$$

$$r2 = [l_2\ cos(\theta 1 + \ \theta 2)\ , l_2\ \ sin(\theta 1 + \ \theta 2)] \quad \textbf{(3.8)}$$

Maka,

$$x = l_1\ cos\ \theta 1 + \ l_2\ \ cos(\theta 1 + \ \theta 2) \quad \textbf{(3.9)}$$

$$z = l_1\ sin\ \theta 1 + \ l_2\ \ sin(\theta 1 + \ \theta 2) \quad \textbf{(3.10)}$$

Persamaan (3.9) dan (3.10) adalah persamaan kinematik maju dari gambar 3.7. Kinematika invers robot dapat dijabarkan sebagai berikut. Dengan menggunakan hukum identitas trigonometri,

$$cos(a+b) = cos(a)\,cos(b) - sin(a)\,sin(b) \quad \textbf{(3.11)}$$

$$sin(a+b) = sin(a)\,cos(b) + sin(b)\,cos(a) \quad \textbf{(3.12)}$$

Maka, didapatkan persamaan baru (3.13) dan (3.14) dari persamaan (3.10) dan (3. 11)

$$\mathrm{x} = l_1\ cos\theta 1 + l_2\ cos\theta 1\, cos\theta 2 - l_2\ sin\theta 1\ sin\theta 2 \quad (3.13)$$

$$\mathrm{z} = l_1\ sin\theta 1 + l_2\ sin\theta 1\, cos\theta 2 - l_2\ cos\theta 1\ sin\theta 2 \quad (3.14)$$

Dari persamaan (3.13) dan (3.14) dapat dicari nilai $\theta 2$ dengan mengeluarkan $cos\theta 1$ dari kedua persamaan, dengan operasi pangkat dua didapatkan,

$$cos\theta 2 = \frac{x^2 + z^2 - {l_1}^2 - {l_2}^2}{2 l_1 l_2} \quad (3.15)$$

Sehingga nilai $\theta 2$ adalah,

$$\theta 2 = arccos\frac{x^2 + z^2 - {l_1}^2 - {l_2}^2}{2 l_1 l_2} \quad (3.16)$$

Sudut $\theta 1$ dapat dicari melalui, perhatikan gambar 3.6

$$\alpha = arctan\frac{l_2\, sin\,\theta 2}{l_2\, cos\,\theta 2 + l_1}, dan \quad (3.17)$$

$$\beta = arctan\frac{\mathrm{z}}{\mathrm{x}} \quad (3.18)$$

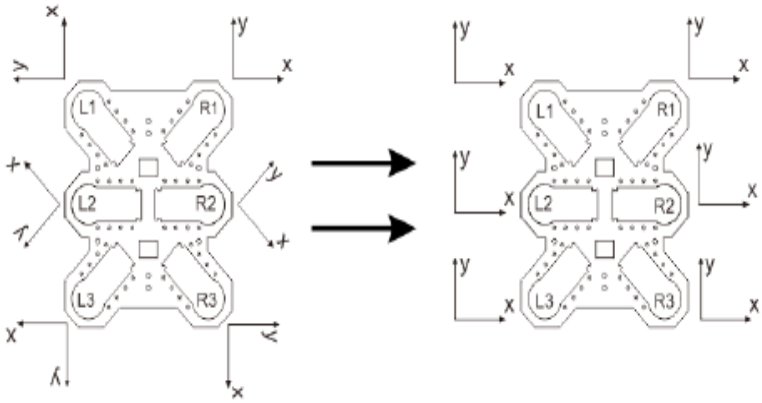
Sedangkan nilai $\theta 1\ adalah,$

$$\theta 1 = \beta - \alpha \quad (3.19)$$

3.3 Konstruksi Badan Robot *Hexapod*

Robot *hexapod linxmotion* pada gambar 3.8 termasuk jenis robot *hexapod* yang memiliki tipe persegi panjang dengan kaki-kaki berada pada dua sisi yang berlawanan. Tipe seperti ini memiliki keunggulan

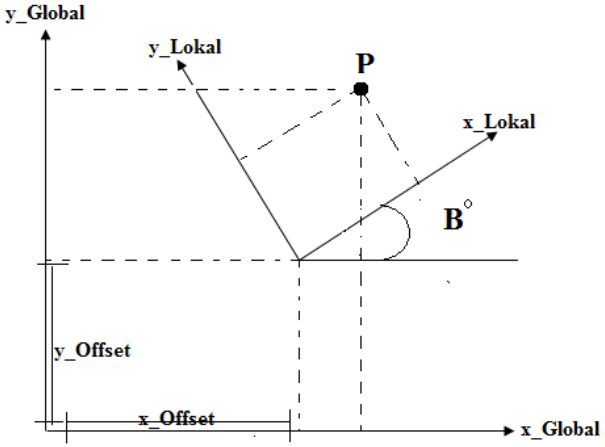
dapat bergerak cepat pada gerakan lurus namun kurang fleksibel untuk gerak berputar.



Gambar 3.8 Konstruksi Badan dan Kaki Robot

Dimensi dan kontruksi robot dapat dilihat pada gambar 3.7. badan robot memiliki panjang 150 mm dan lebar 60 mm. Posisi kaki pada ujung badan (kaki 1,2,5,6) membentuk sudut sebesar -20 derajat dan 20 derajat saat kondisi normal atau *stand by*. Selain itu posisi ini adalah sebagai posisi tengah dari gerakan langkah kaki.

Koordinat kaki diukur dari lokal koordinat yang digunakan untuk mengkoordinasikan masing-masing kaki dengan merepresentasikan koordinat lokal ke koordinat global pada titik tengah badan robot, dapat dilihat pada gambar 3.9.



Gambar 3.9 Representasi Koordinat Lokal dan Koordinat Global

Koordinat lokal pada kaki (1,2,5,6) mengalami translasi sebesar (x_offset, y_offset) dan rotasi pada sumbu z sebesar α°, maka dapat ditulisakan persaman matriks homogen sebagai berikut, dengan nilai z yang tidak berubah:

$$\begin{bmatrix} Px \\ Py \\ Pz \\ 1 \end{bmatrix} = \begin{bmatrix} cos\beta & sin\beta & 0 & x\_offset \\ -cos\beta & -sin\beta & 0 & y\_offset \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Px\_local \\ Py\_local \\ Pz\_local \\ 1 \end{bmatrix} \quad (3.20)$$

Nilai x_offset, y_offset dan sudut α° tiap kaki tidak sama. Tabel 3.1 menunjukkan parameter dari tiap masing - masing kaki yang akan digunakan untuk mencari koordinat lokal untuk mengkoordinasi pergerakan masing – masing kaki.

Tabel 3.2 Koordinat Base Frame Kaki Robot

Kaki	X_offsset	Y_offsest	B(°)
Kaki 2	30	70	20
Kaki 4	50	0	0
Kaki 6	30	-70	-20

3.4 *Trayectori* Planning

Trayectori planning sangat penting diperlukan untuk membuat pola berjalan sehingga robot dapat berjalan sesuai dengan jalur *trayectori* yang dibuat. *Trayectori* yang akan digunakan pada robot *hexapod* ini adalah dengan menggunakan polinomial derajat satu dengan nilai pada sumbu y sebagai nilai polinomial derajat satu.

$$y(t) = a_0 + a_1 t \tag{3.21}$$

$$y'(t) = a_1 \tag{3.22}$$

Nilai sumbu y di tuliskan dengan y(t) yaitu posisi sumbu y pada saat t, t dalam satuan waktu yang bisa di tuliskan sebagai iterasi y'(t) adalah kecepatan perubahan posisi tiap iterasi sebesar a1, jadi kecepatannya memiliki kecepatan yang tetap. Nilai pada sumbu y diambil karena perubahan posisi pada sumbu y menyebabkan robot berjalan lurus maju. Nilai pada sumbu x bisa dihitung dengan,

$$x(t) = \mathrm{y(t)} \tan\gamma \tag{3.23}$$

$$z(t) = Fc\ (\pi \frac{t - t_0}{tf - t_0}) \tag{3.24}$$

$\gamma = sudut\ kemiringan\ robot\ (berjalan\ lurus = 0)$

$F_c = Amplitudo\ titik\ puncak\ trayectori$

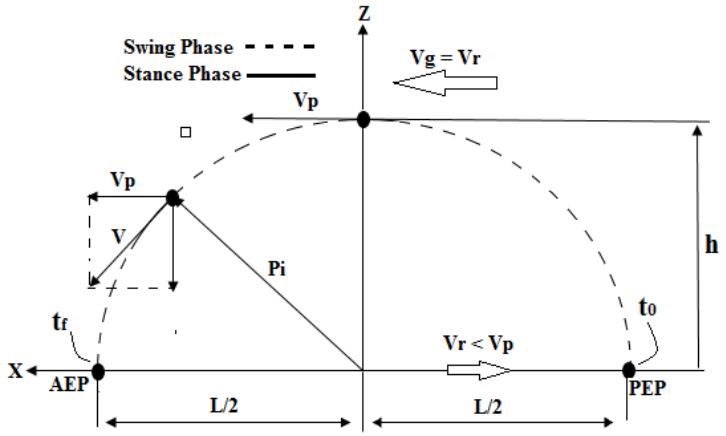
$t_0 = Waktu\ awal\ fase\ transfer$

$t_f = Waktu\ awal\ fase\ support$

$\mathrm{t} = Waktu\ saat\ ini$

$a_0 = posisi\ awal\ sumbu\ y$

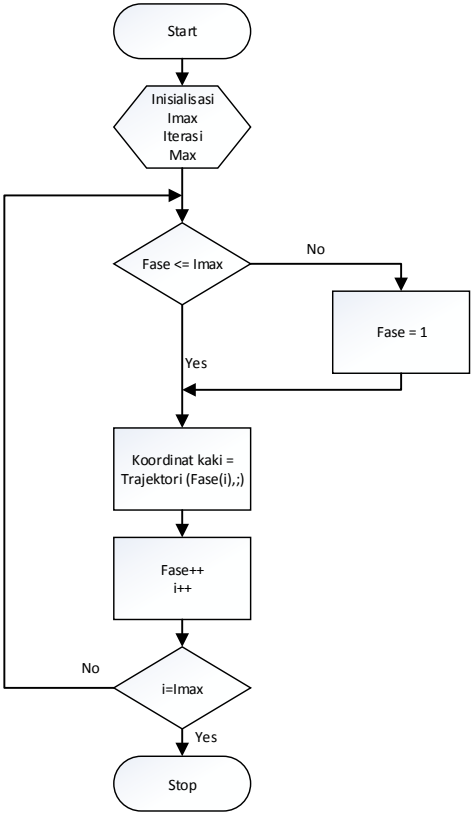
$a_1 = posisi\ tujuan\ sumbu\ y$



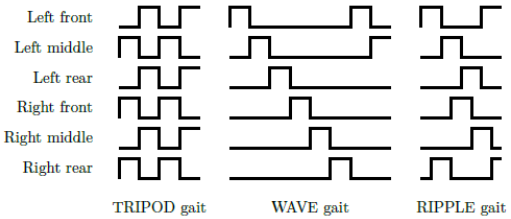
Gambar 3.10 *Trayectori Planning* Kaki Robot *Hexapod*

3.4.1 *Gait* Robot *Hexapod* Jalan Lurus

Gait adalah gaya berjalan robot gaya berjalan robot *hexapod* yang paling stabil[XX] adalah jenis *tripod* dengan 3 kaki menginjak tanah dan 3 kaki melayang yang pada kedua sisinya saling berbeda. Pada *gait* berjalan lurus *trayectori* jalur pada ujung kaki robot, *trayectori* hanya mencakup untuk jalur pada satu kaki robot sedangkan gait adalah metode untuk mengkoordinasikan enam kaki robot agar robot bergerak sesuai jalurnya. Gambar 3.11 adalah *flowchart* algoritma pembangkitan gait pada robot *hexapod.*



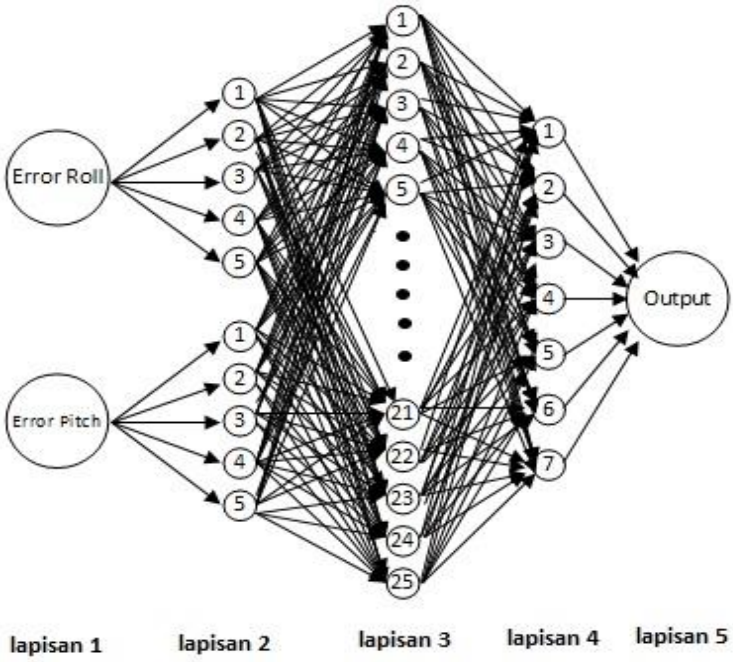
Gambar 3.11 Algoritma *Gait* Robot *Hexapod*



Gambar 3.12 Jenis *Gait Hexapod*

3.5 Perancangan Kontroler *Neuro-Fuzzy*

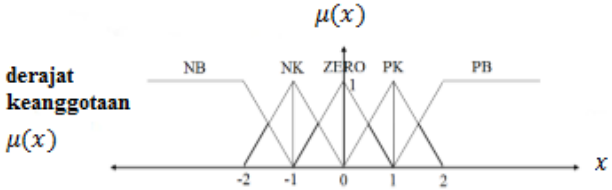
Neuro-fuzzy merupakan metode kontrol yang berpedoman pada logika *fuzzy logic* dengan tambahan algoritma pembelajaran *neural network* sehingga nilai paramater – parameter pada logika *fuzzy* dapat berubah sesuai dengan nilai pembelajaran dari *neural network*. Pada penelitian kali ini metode ini digunakan untuk mencari posisi dari robot apabila robot di rotasi terhadap sumbu x dan y output dari kontroler ini berupa nilai posisi yang sesuai dengan kemiringan badan robot sehingga didapatkan badan robot yang tetap seimbang walaupun di beri gangguan berupa sudut rotasi sumbu x dan y. Alur metode ini sama dengan logika *fuzzy* dengan 5 layer, yang telah dijelaskan pada bab 2, dengan menggunakan logika mamdani (*min, max*).



Gambar 3.13 Struktur Neuro Fuzzy

1. *Fuzzyfikasi*

   Fuzzyfikasi yang digunakan adalah menggunakan 5 himpunan pendukung dengan masing-masing 5 himpunan pendukung untuk *error* pada sudut *roll* dan *error* pada sudut *pitch.* Himpunan pendukung yang digunakan pada penelitian ini adalah berbentuk segitiga yang ditunjukkan pada gambar 3.14



Gambar 3.14 Himpunan *Fuzzy Error Roll* dan *Pitch*

2. *Rule Base* dan *Inferensi Fuzzy*

   Rule base adalah layer untuk pemberian nilai aturan (*min*, *max*) pada nilai output dari layer 1 (*fuzzyfikasi*), sebagai contoh "jika error roll negatif besar dan error pitch positif besar maka sinyal kontrol posisi adalah positif kecil". *Rule base* mengambil peran pada kata "dan" yang artinya adalah operasi minimum. Sedangkan inferensi adalah proses pengambilan kesimpulan dari aturan rule base dan proses fuzzyfikasi sehingga pada contoh bisa di sebut dengan kata "maka" yang artinya adalah suatu akibat dari sebab (rule base dan fuzzyfikasi. Dengan cara ini pemetaan tingkatan menjadi lebih mudah untuk dipahami dan mudah untuk di logika. Pada penelitian kali ini menggunakan rule base mack vicar wheelan yang menuliskan rule base dalam bentuk sebuah matriks 2 dimensi yang sama dan yang berdimensi sama seperti himpunan pendukung. Contoh rule base mack vicar wheelan dan inferensinya bisa dilihat pada tabel 3.3.

Tabel 3.3 Rule Base Mack Vicar Wheelan dan Inferensi Fuzzy (Kaki no. 1)

	Error Pitch				
Error roll	6	6	6	7	7
	4	5	6	6	7
	2	3	4	5	6
	4	3	2	2	1
	1	1	2	2	2

3. Normalisasi
   Normalisasi ini berada pada layer 3 yang berfungsi untuk menormalkan atau menjadikan semua output pada layer 2 sehingga semua output pada layer 2 memiliki nilai yang jika di jumlahkan adalah 1, hal ini dikarena untuk menjaga agar nilai penjumlahan layer 2 tidak lebih dari 1. Contoh algoritma normalisasi bisa dilihat pada persamaan 3.25

$$\overline{w}_i = \frac{w_i}{w_1 + \cdots + w_n} \tag{3.25}$$

   n = jumlah himpunan output fuzzy
   $\overline{w}_i$ = output ternormalisasi, ke-i s/d n

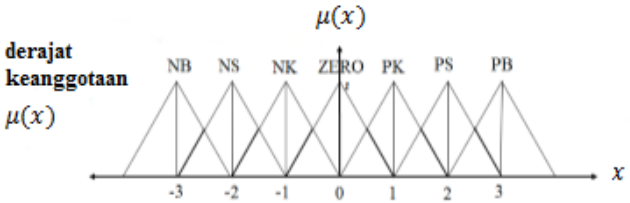
4. Operasi Nilai *Consequent Paramater*
   Nilai *consequent parameter* adalah himpunan nilai untuk untuk melakukan proses revisi pada layer 4 pada layer yang merupakan himpunan nilai *center of area* untuk proses *defuzzyfikasi*, sehingga apabila melakukan revisi pada layer ini maka nilai *center of area* himpunan output *fuzzy* akan bergeser semakin besar atau semakin kecil. Sehingga kontroler *neuro fuzzy* mampu melakukan *learning* (pembelajaran). Persamaan *consequent parameter* dapat dilihat pada persamaan (3.26). nilai f_i awal adalah nilai *random* pada proses inisialisasi

$$\overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i) \quad (3.26)$$

f_i = *Consequent Parameter*

5. *Defuzzyfikasi*
   Defuzzyfikasi merupakan proses pengembalian dari nilai logika *fuzzy* menjadi logika aslinya proses yang digunakan dalam operasi ini adalah *center of gravitty* yang nilai himpunan *center of gravity* telah di tentukan pada layer 4 dan himpunan *center of gravity* dapat berubah unutuk melakukan *learning*. Pada penelitian ini digunakan 7 himpunan *output* yang bisa dilihat pada gambar 3.15, sedangkan persamaan defuzzyfikasi output dapat dilihat pada persamaan (3.27)

$$\sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (3.27)$$



Gambar 3.15 *Defuzzyfikasi*

6. *Back Propagation* atau Revisi Nilai Parameter
   Perhitungan nilai error dilakukan dengan cara *direct* yaitu nilai error yang di dapat dari output neuro-fuzzy dibandingkan dengan nilai output dari referensinya atau yang diharapkan. Revisi bobot dihitung dengan persamaan (3.28)

   $fi = f(i-1) + lr * e * Wi$ (3.28)

   fi = *consequent parameter* baru

   f(i-1) = *consequent paramter* lama

   lr = *learning rate*

e = *error* (*Reference* – output defuzzyfikasi)

Wi = *output* ternormalisasi

Dari keenam proses neuro-fuzzy nilai output yang didapatkan adalah besarnya posisi sumbu z pada koordinat global badan robot apabila terjadi rotasi roll dan pitch. Kemudian koordinat global yang didapat di proyeksikan kedalam koordinat lokal pada setiap kaki untuk dilakukan proses invers kinematik sesuai dengan proyeksi koordinat tersebut. Ada 2 mode dalam kontrol keseimbangan robot hexapod yaitu mode kontrol keseimbangan dengan posisi robot diam dan kontrol keseimbangan dengan posisi robot berjalan.

BAB IV
PENGUJIAN DAN ANALISA

Bab ini menjelaskan hasil pengujian dan analisa mengenai rancangan penelitian yang sudah dibuat. Pengujian dan analisa yang dilakukan tersebut terkait dengan pengujian masing – masing komponen dari sistem sampai performansi sistem secara keseluruhan.

4.1 Pengujian Mekanik Robot

Bagian mekanik robot ini merupakan komponen yang telah di desain sebelumnya dan telah dilakukan uji coba pada desain yang telah dibuat sehingga sudah sampai pada pasar. Robot *hexapod* pada penelitian kali ini memiliki bagian mekanik yaitu dua bagian badan atas dan bawah, dua belas penyangga servo poros yang berhubungan dengan badan atas (*coxa*), enam bagian batang kaki tengah (*femur*), dan enam bagian penyangga akhir atau kaki tumpuan robot terhadap tanah (*tibia*).

4.1.1 Pengujian Motor *Servo*

Pengujian motor *servo* MG995 dilakukan dengan memberikan tegangan input dan memberikan data sudut motor kemudian sudut putaran dari *horn servo.* Sudut yang keluaran yang dihasilkan akan diukur dengan menggunakan busur derajat dan data sudut terukur dibandingkan dengan data sudut masukan yang diberikan untuk mengukur tingkat keakuratan motor *servo* MG995.

Dari data yang diperoleh kemudian dihitung nilai *RMSE (Root Mean Square Error)* dengan persamaan (4.1).

$$RMSE = \sqrt{\frac{\sum_{i=0}^{k}(\mathrm{y} - y_i)^2}{k}} \tag{4.1}$$

Nilai ini digunakan sebagai nilai yang akan digunakan untuk menghitung nilai *Full Scale Error* (FSE). Persamaan nilai FSE dapat dilihat pada persamaan (4.2)

$$FSE = \frac{\text{RMSE}}{Full\ Scale} x\ 100\% \quad (4.2)$$

Data haisl pengujian sudut motor servo MG995 ditunjukkan pada tabel (4.1) dibawah,

Tabel 4.1 Data Pengujian Motor Servo

Sudut Acuan (°)	Sudut Terukur (°)	Error (°)
0	0	0
10	10	0
20	20.2	-0.2
30	30.1	-0.1
40	40	0
50	50.4	-0.4
60	60.2	-0.2
70	70.5	-0.5
80	80.3	-0.3
90	90	0
100	100	0
110	110.6	-0.6
120	120.1	-0.1
130	130	0
140	140.4	-0.4
150	150	0
160	160	0
170	170.7	-0.7
180	180	0

Berdasarkan data pada tabel (4.1) nilai error diolah dengan menggunakan RMSE sehingga didapatkan tingkat kesalahan sebesar 0,184211° dan FSE sebesar 0,102339%, sehingga memiliki keakuratan sebesar 0,102339%. dari pengambilan data dihasilkan nilai error yang kecil sehingga bisa disimpulkan bahwa motor servo memiliki tingkat keakuratan yang cukup tinggi, sehingga cocok digunakan sebagai aktuator dari robot *hexapod*.

4.1.2 Pengujian *Invers* Kinematika Robot

Pengujian invers kinematik robot dilakukan untuk mengukur tingkat kebenaran dari algoritma persamaan yang sudah dibuat. Pengujian ini dilakukan dengan memberikan nilai input posisi robot pada sumbu x, y dan z dan mengukur posisi robot menggunakan penggaris kemudian data input posisi di bandingakan dengan data hasil pengukuran. Pengukuran ini dibedakan menjadi 3 tahap yaitu pengujian terhadap posisi x, posisi y dan posisi z. Tabel 4.2 merupakan hasil pengujian terhadap sumbu x, tabel 4.3 merupakan hasil pengujian terhadap sumbu y, dan tabel 4.4 merupakan hasil pengujian terhadap sumbu z. Nilai z_*offset* sebesar 100 mm nilai ini adalah nilai posisi default kaki robot saat bernilai posisi z = 0

Tabel 4.2 Pengujian *Invers* Kinematik terhadap sumbu x

***Input* Posisi (mm)**	***Output* Posisi (mm)**	**Error (mm)**
100	102,6	-2,6
230	233,5	-3,5
90	89,5	0,5
114	112,7	1,3
56	57.1	-1,1
0	0,3	-0,3
150	153,0	-3,0
70	69,6	0,4

Tabel 4.3 Pengujian *Invers* Kinematik terhadap sumbu y

***Input* Posisi (mm)**	***Output* Posisi (mm)**	**Error (mm)**
0,0	0,0	0
20	20,5	-0,5
40	43,2	-3,2
65	66,7	-1,7
-20	-20	0
-30	-30,3	-0,3
-49	-50,1	-1,1
30	30,0	0

Tabel 4.4 Pengujian *Invers* Kinematik terhadap sumbu z

***Input* Posisi (mm)**	***Output* Posisi (mm)**	**Error (mm)**
0,0	0,0	0
10	10,2	-0,2
40	40,1	-0,2
-20	-20,3	-0,3
30	30	0
-50	-49,8	0,2
20	20,4	-0,4
60	60	0

Dari hasil pengujian tersebut terdapat *error* rata – rata pada posisi sumbu x sebesar 1,5875 mm, pada sumbu y sebesar 0,85 mm, dan pada sumbu z sebesar 0,1625 mm. Kesalahan *output* posisi terjadi karena nilai pengukuran yang kurang presisi terhadap panjang *link* pada kaki robot dan komputasi perhitungan yang terbatas karena perhitungan komputasi trigonometri sehingga menyebabkan *error* pada nilai *ouput* posisi robot, tingkat keakuratan *servo* juga berpengaruh pada nilai *error* ini karena motor terbebani oleh berat robot. Hal ini yang menyebabkan hasil *output* posisi terdapat *error*. Dari data pengujian dapat disimpulkan bahwa kesalahan posisi terdapat pada sumbu z dengan dua digit nilai error dalam (mm).

4.1.3 Pengujian *Trayectori Planning*

Trajektori kaki bekerja pada sumbu x, y dan z tetapi untuk gerakan lurus maju pada sumbu y maka sudut $\gamma = 0$ sehingga hanya bekerja pada sumbu y. Untuk menghitung nilai pada sumbu x dengan cara memperhitungkan nilai γ sudut serong. Untuk lebih memudahkan analisa perancangan trayektori dibedakan menjadi 2 tahap yaitu pada sumbu x-y lalu pada bidang y-z. Trayektoi yang digunakan adalah polinomial orde satu untuk bidang x dan y. Persamaan (4.3) dan (4.4) adalah persamaan posisi pada sumbu y.

$$y(t) = a_0 + a_1 t \quad 4.3$$

$$y'(t) = a_1 \quad 4.4$$

Variable y adalah posisi titik – titik pada sumbu y pada saat waktu ke-t dan y' adalah kecepatan perpindahan tiap titik – titik pada sumbu y sebesar a1. Nilai posisi x(t) dicari menggunakan persamaan (4.5).

$$x(t) = y(t)\ tan\ (\gamma) \qquad 4.5$$

Berikutnya untuk menghitung nilai pada tahap kedua adalah pada bidang y-z pada persamaan (4.6).

$$z(t) = \begin{cases} Fc \sin\left(\pi \dfrac{t - t_0}{t_{f-}\, t_0}\right), & transfer\ phase \\ 0, support\ phase \end{cases} \qquad 4.6$$

Fc = Amplitudo Kurva sumbu z

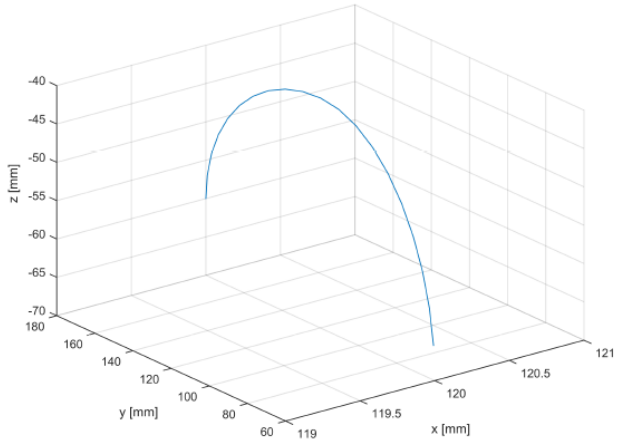
t = iterasi

t_0= waktu (t) pada saat awal fase *support*

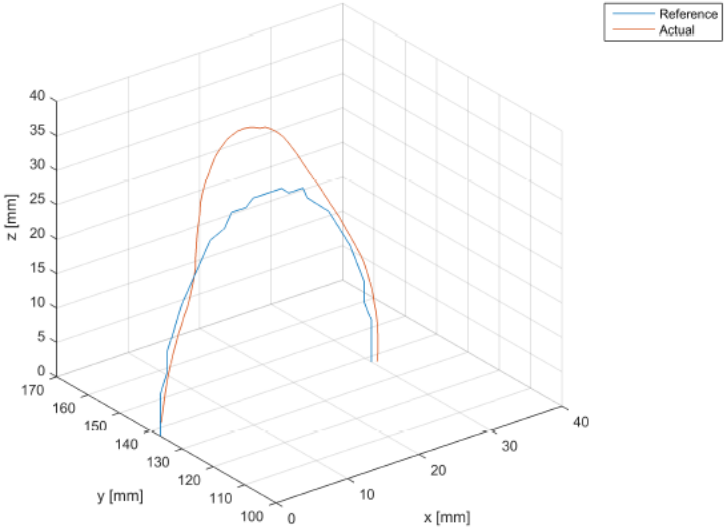
t_f= waktu (t) pada saat awal fase *transfer*

Waktu t adalah iterasi dengan Δt tertentu yaitu dalam satuan detik iterasi menentukan banyaknya titik – titik yang akan dilalui untuk berpindah dari fase *support* ke fase *transfer* dan sebaliknya dalam satu iterasi adalah satu siklus yaitu fase *support* – fase *transfer* – fase *support*. Semakin banyak iterasi maka semakin halus kurva yang dihasilkan dan semakin lama waktu yang dibutuhkan dan juga komputasinya akan semakin banyak.apabila iterasi kurang atau terlalu sedikit maka kurva yang dihasilkan juga kasar dan terlihat kaku. Gambar (4.1) adalah gambar grafik kurva simulasi pembangkitan trajektori.

Untuk melihat kurva hasil implemetasi dibutuhkan penanaman program pada kontroler robot dan dilakukan proses perbandingan dengan kurva hasil simulasi program, gambar (4.2) adalah gambar kurva antara simulasi dan implementasi



Gambar 4.1 Kurva Hasil Simulasi Trajektori

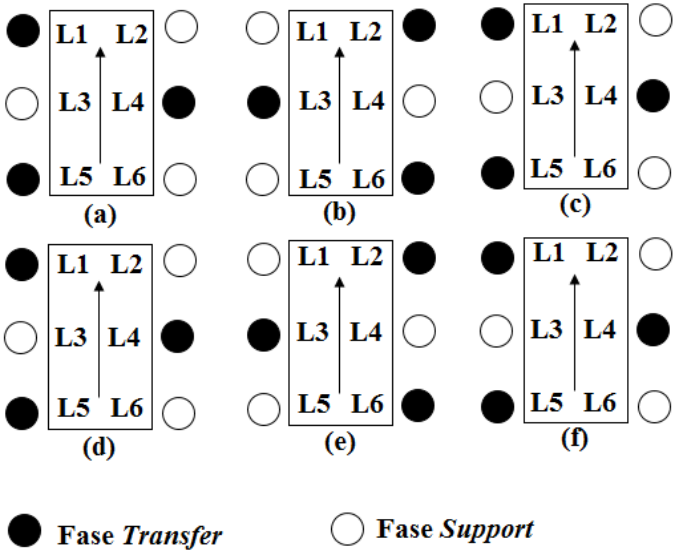


Gambar 4.2 Kurva Hasil Simulasi dan Implementasi

Dari hasil simulasi dan implementasi pada gambar (4.2) terdapat selisih antara nilai puncak dari kedua kurva hal ini disebabkan karena posisi z pada setiap kaki robot memiliki nilai yang tidak sama karena perbedaaan hardware servo dan tingkat akurasi servo yang berbeda – beda.

4.1.4 Pengujian Algoritma Gait

Pengujian ini dilakukan dengan cara mencoba algoritma program pada robot secara langsung dengan memperhatikan pola langkah tiap kaki pada robot algoritma yang digunakan adalah jenis tripod gait. Gambar (4.3) merupakan pola langkah tripod gait.



Gambar 4.3 Urutan Langkah Pola Tripod Gait

Pada gambar (4.3) didapatkan bahwa mode ini melangkahkan 3 kaki dengan pola segitiga jika dilihat pada sumbu z. Pergerakan kaki jenis ini hanya membutuhkan dua iterasi untuk kembali pada posisi awal titik beban berada pada tengah badan robot akan membuat pola ini memiliki tingkat kestabilan statis yang tinggi.

4.2 Pengujian Perangkat Elektronika Robot

Pengujian ini dilakukan untuk mengetahui kemampuan kerja dari semua perangkat elektronika robot yang digunakan pada penelitian ini. Pada pengujian ini akan dilakukan pengujian sensor IMU dan perangkat komunikasi.

4.2.1 Pengujian *IMU*

Sensor posisi yang digunakan pada penelitian ini adalah MPU6050. Pengujian sensor posisi *IMU* dibagi menjadi dua tahap pengujian yaitu pengujian *roll* dan *pitch*. Sebelumnya telah dilakukan inisialisasi awal dengan meletakkan sensor sampai proses inisialisasi selesai. Pengujian ini dilakukan dengan menggunakan busur derajat yang di sesuaikan dengan posisi sensor IMU. Nilai yang akan menjadi perbandingan adalah sudut aktual dengan output sensor untuk menguji tingkat keakuratan dari sensor MPU6050. Tabel 4.5 merupakan data pengujian terhadap sudut roll dan Tabel 4.6 merupakan data pengujian terhadap sudut pitch.

Tabel 4.5 Hasil Pengujian MPU6050 terhadap sudut roll

	Nilai Output Sensor				
Pengambilan Data	**1**	**2**	**3**	**4**	**5**
Sudut Aktual					
0	0,62	0,62	0,5	0,62	0,19
10	10,15	10,24	10,32	10,49	10,16
20	20,72	20,51	20,22	20,51	20,25
30	30,51	30,22	30,36	30,62	30,21
40	40,42	40,37	40,43	40,12	40,33
50	50,21	50,51	50,22	60,15	50,32
60	60,23	60,72	60,45	60,05	60,15
70	70,2	70,52	70,19	70,42	70,13
80	80,32	80,41	80,62	80,46	80,54
90	88,67	89,05	89,43	89,24	88,25

Berdasarkan data pada tabel 4.5 didapatkan rata – rata nilai kesalahan output sensor MPU6050 adalah sebesar 0,308 derajat pada

sudut *roll* dihitung dengan RMSE sensor ini termasuk sensor yang linier.

Tabel 4.6 Hasil Pengujian MPU6050 terhadap sudut pitch

	Nilai Output Sensor				
Pengambilan Data	**1**	**2**	**3**	**4**	**5**
Sudut Aktual					
0	0,44	0,23	0,11	0,53	0,22
10	10,62	10,34	10,39	10,01	10,47
20	20,13	20,28	20,42	20,15	20,66
30	30,03	30,59	30,39	30,52	30,63
40	40,19	40,42	40,33	40,37	40,73
50	50,56	50,51	50,22	50,52	50,36
60	60,33	60,72	60,63	60,32	60,52
70	70,23	70,52	70,16	70,61	70,51
80	80,98	80,41	80,41	80,53	80,37
90	89,67	89,34	90,02	89,21	89,45

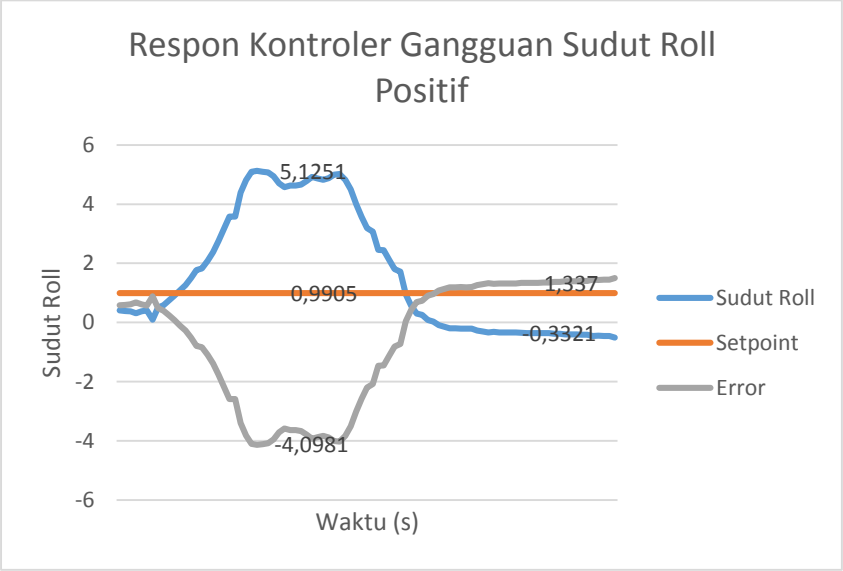
Berdasarkan data pada tabel 4.5 didapatkan rata – rata nilai kesalahan output sensor MPU6050 adalah sebesar 0,403 derajat pada sudut p*itch* dihitung dengan RMSE sensor ini termasuk sensor yang linier. Dengan data yang didapat pada tabel 4.5 dan tabel 4.6 sensor MPU6050 cocok digunakan pada robot *hexapod*.

4.3 Pengujian Kontroler Neuro - Fuzzy

Pengujian kontroler *neuro – fuzzy* pada robot *hexapod*. Kemudian diambil data pengukuran dengan pengambilan data menggunakan radio. Data hasil pengukuran kemudian dianalisa, pengujian ini dilakukan dengan memberikan gangguan berupa kemiringan posisi dari pijakan kaki robot dibagi menjadi dua tahap yaitu pada sudut *roll* dan sudut *pitch*. Pengujian kontroler pada sudut roll dapat dilihat pada gambar (4.4).

4.3.1 Pengujian Kontroler Terhadap Gerakan *Roll*

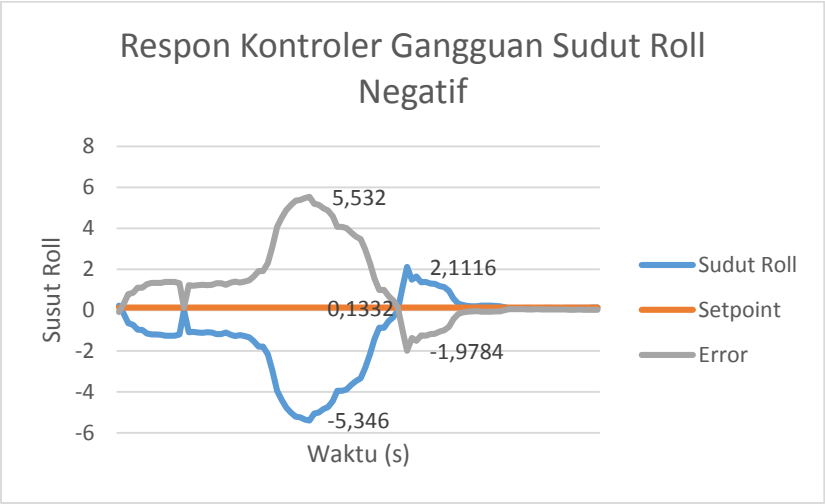
Pengujian gerakan terhadap sudut *roll* dilakukan dengan memberikan kemiringan tanah tempat robot berpijak dari kemiringan 0, lebih besar dari 0 (*roll* positif), kemudian sampai kurang dari 0 (*roll* negatif). Sudut kemiringan tanah di tahan dalam beberapa detik untuk melihat respon robot secara bertahap. Gambar grafik respon robot pada kemiringan roll positif dapat dilihat dari gambar (4.4).



Gambar 4.4 Grafik Respon Kontroler Gerakan *Roll* Dengan Kemiringan 5,1251 derajat.

Pada gambar (4.4) nilai setpoint kemiringan didapatkan nilai setpoint kemiringan sebesar 0,9905° didapatkan saat proses inisialisasi awal dengan mencari posisi dari setiap kaki kemudian menghitung seberapa besar simpangan yang dihasilkan. Kemudian secara perlahan kontroler akan menyesuaikan dengan nilai referensi yang di berikan.algoritma pembelajaran dan logika *fuzzy* mengakibatkan nilai dari respon robot menjadi sesuai dengan referensinya. Kemudian

secara bertahap tanah di miringkan sebesar 5,1251 derajat. Selanjutnya dilakukan pengembalian posisi dari tanah. Dari data diatas respon robot menunjukkan dapat mempertahankan posisinya dimana di putar *roll* dengan error sebesar -0,3321 derajat.



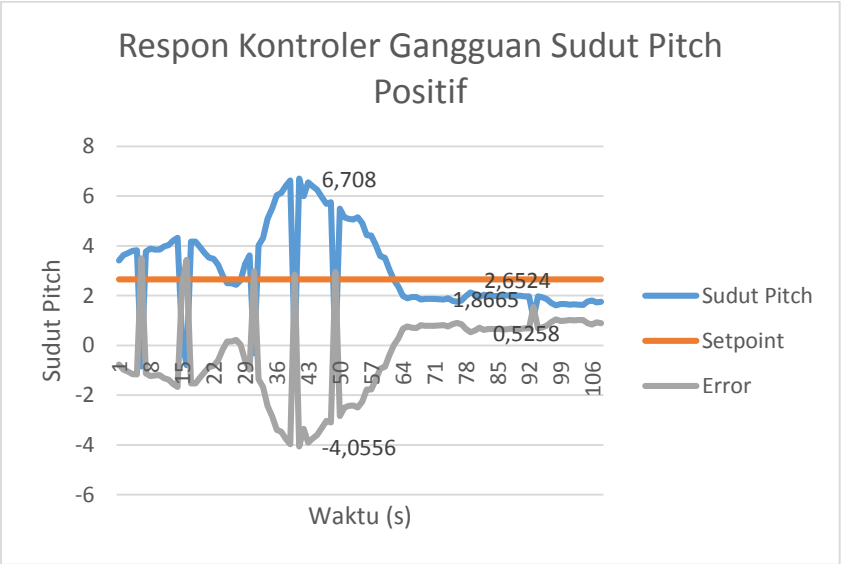
Gambar 4.5 Grafik Respon Kontroler Gerakan *Roll* Dengan Kemiringan -5,346 derajat.

Pada grafik gambar (4.5) menunjukkan nilai *error* keadaan *steady state* sekitar mendekati nilai setpointnya 0.1332 derajat, akan tetapi terjadi *overshoot* sebesar 2,1116 derajat. Hal ini disebabkan karena pengembalian posisi pijakan awal robot sehingga robot menyeimbangkan kembali. Gambar grafik (4.5) menunjukkan bahwa kontroler mampu menyeimbangkan posisi nya kembali.

4.3.2 Pengujian Kontroler Terhadap Gerakan *Picth*

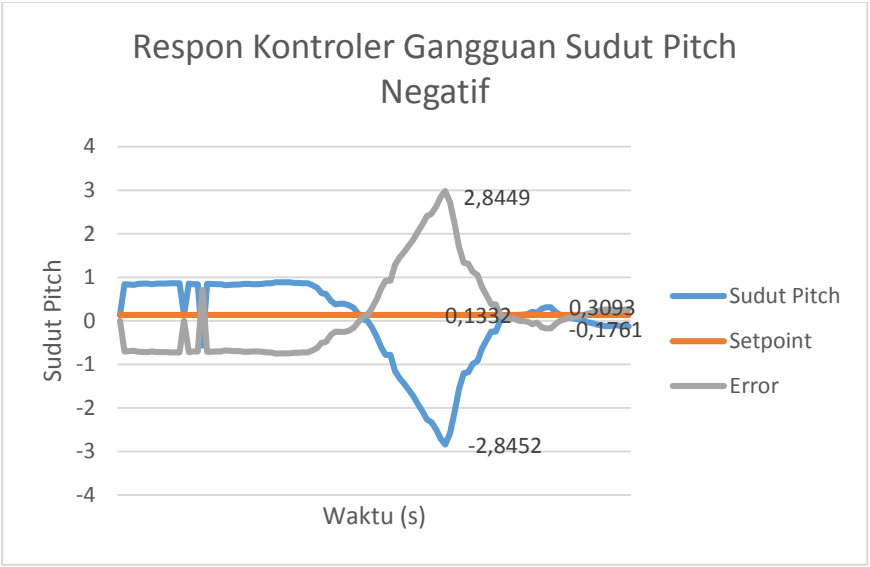
Pengujian respon robot terhadap gerakan *pitch* dilakukan sama seperti gerakan *roll* yaitu dengan memberikan kemiringan yang positif dan negatif bertahap pada tanah tempat pijakan robot namun bedanya terletak pada sumbu yang diputar yaitu pada sumbu y. Proses

mya sama seperti gerakan *roll* memberikan nilai kemiringan pada sudut kurang dari 0 derajat (*pitch* negatif) dan lebih dari 0 (*pitch* positif). grafik pengujian respon robot pada gerakan pitch dengan nilai sudut kemiringan 6,708 derajat dapat dilihat pada gambar (4.6).



Gambar 4.6 Grafik Respon Kontroler Gerakan *Pitch* Dengan Kemiringan 6,708 derajat.

Pada gambar (4.6) dapat dilihat bahwa kemiringan robot mengalami kenaikan bertahap sampai dengan sekitar 6,708 derajat kemudian secara bertahap dengan waktu yang berbeda grafik respon nya turun sampai 1,8665 derajat kemudian mencapai *steady state* pada nilai teresebut. Dengan nilai *steady state* 1,8665 didapatkan nilai *error* sebesar 0,5258 derajat. Grafik respon diatas menunjukkan bahwa terdapat data sudut pembacaan yang tidak terfilter, hal ini menyebabkan grafik respon mengalami fluktuasi yang tinggi. gambar grafik (4.6) menunjukkan bahwa kontroler dapat menyembangkan posisi nya dengan *error* sebesar 0,5258 derajat.



Gambar 4.7 Grafik Respon Kontroler Gerakan *Pitch* Dengan Kemiringan -2.8452 derajat.

Berdasarkan gambar (4.7) respon robot tidak banyak berubah dari kondisi sebelumnya robot mampu menyeimbangkan posisi nya sesuai dengan setpoint yang di berikan. Pada kondisi *steady state* nilai *error* nya yaitu -0,1761. Respon pada gangguan pitch pada saat kondisi steady state mengalami osilasi. Hal ini bisa dikarenakan oleh kondisi robot yang kurang kokoh sehingga menyebabkan perubahan nilai sudut yang terbaca.

-- Halaman ini sengaja dikosongkan --

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan pengujian dan analisa dapat diambil kesimpulan bahwa :

1. perubahan posisi pada sumbu x pada *invers* kinematik mempengaruhi pergerakan *joint femur* dan *tibia*, perubahan posisi pada sumbu y pada *invers* kinematik mempengaruhi pergerakan pada *joint coxa* dan perubahan posisi pada sumbu z pada invers kinematik mempengaruhi pergerakan pada *joint femur* dan *tibia*.
2. Nilai RMSE pada *invers* kinematik masing – masing adalah 1,5875 mm pada sumbu x, 0,85 mm pada sumbu y, dan pada sumbu z sebesar 0,1625 mm.
3. Tingkat akurasi dari motor servo dan *driver*-nya sangatlah penting untuk membuat robot dapat memiliki perhitungan error yang kecil.
4. Torsi dan reolusi motor servo yang digunakan sangatlah terbatas sehingga menghasilkan nilai *error*.
5. Gangguan pada sudut *roll* tidak terlalu bermasalah terhadap kestabilan robot pada bidang miring, sedangkan pada sudut *pitch* akan berpengaruh pada orientasi *heading* robot yang memiliki respon dengan osilasi.
6. Kontroler logika *fuzzy* dengan *neural netwrok* pada penelitian ini menggunakan kontroler tipe P (*proportional*) pada robot akan menyebabkan osilasi posisi seimbang robot jika nilai parameter *gain input* dan *output* tidak sesuai.
7. Kontroler ini juga memiliki nilai *error* pada gangguan *roll* positif sebesar (-0,3321 derajat), *error* gangguan *roll* negatif sebesar (0,1342 derajat), error gangguan *pitch* positif sebesar (0,5258 derajat) dan *error* gangguan *pitch* negatif sebesar (-0.1761 derajat).
8. Apabila nilai *gain input fuzzyfikasi* di besarkan maka kontroler akan menambah ketelitian nilai *output* tetapi

respon akan menjadi sedikit lebih lambat, dan sebaliknya, hal yang sama terjadi pada g*ain output defuzzyfikasi*.

9. Nilai *output* dari sensor MPU6050 terkadang mengalami gangguan yang menyebabkan nilai nya menjadi sangat besar.
10. Konstruksi kaki robot pada bagian femur yang digunakan kurang kokoh sehingga menyebabkan salah satu dari kaki terjadi pergeseran ini menyebabkan sudut yang terbaca juga akan berubah.

5.2 Saran

1. Melakukan perhitungan yang tepat dan sesuai untuk menghitung nilai *invers* kinematik dan *forward* kinematik.
2. untuk lebih tepat dalam perhitungan trigonometri disarankan menggunakan atan2 karena memiliki nilai yang berbeda jika diberi nilai positif dan negatif.
3. Disarankan agar menggunakan *driver* motor servo dan motor servo dengan tingkat akurasi tinggi.
4. Disarankan agar menggunakan kontroler atau *mainboard* sistem yang memiliki spesifikasi tinggi karena diperlukan untuk menghitung sudut sehingga perlu ketelitian tinggi.
5. Untuk kontrol keseimbangan di sarankan agar tidak hanya menggunakan kontroler tipe P (*proportional*), bisa menggunakan PI, PD, maupun PID.
6. Melakukan uji coba secara langsung untuk *setting* nilai parameter *gain input* maupun *output* kontroler ini.
7. *Filter* data sangatlah di perlukan untuk membatasi nilai keluaran dari sensor MPU6050, atau bisa menggunakan sesnor lain yang lebih presisi.
8. Disarankan agar menggunakan konstruksi kaki robot yang kokoh sehingga motor servo tidak bekerja terlalu memaksa hal ini menyebabkan kerusakan pada motor servo dan sudut yang terbaca tidak bergeser.
9. Disarankan agar menggunakan servo dengan tingkat resolusi tinggi untuk mengurangi nilai *error* pada perhitungan *invers* kinematika dan kondisi terbaca.

DAFTAR PUSTAKA

[1] Woering R., *Simulating the "first step" of walking hexapod robot*, Master's Thesis, University of Technology Eindhoven, 2011.

[2] Figliolini G., Stan S D., Rea P., *Motion Analisys of Leg Tip of a Six-Legged Walking Robot*, IFToMM World Congress, 2007.

[3] Pitowarno Endra, *Robotika Desain, Kontrol dan Kecerdasan Buatan*, Andi Offset, Yogyakarta, 2006.

[4] Gouda Bhanu K., *Optimal Robot Trajectory Planning Using Evolutionary Algorithms*, Master's Thesis, Cleveland State University, 2006.

[5] Mark, W. Spong., "*Robot Dynamic and Control*", John Willey and Sons, 2005.

[6] Pa, P.S., Wu, C.M., 2012, "*Design of a hexapod robot with a servo control and a man-machine interface*," Robotics and Computer-Integrated Manufacturing, 28: 351-358.

[7] Wang, Z., Ding, X., Rovetta, A., Glusti, A., 2011, *Mobility Analysis of the Typical Gait of a Radial Symmetrical Six-Legged Robot*, Mechatronics, 21 (7): 1133-1146.

[8] F. Berardi, M. Chiaberge, E. Miranda, and L.M. Reyneri "*A Walking Hexapod Controlled by a Neuro Fuzzy System*". Torino, Italy: Dipartimento di Elettronica, Politecnico

[9] Siciliano, Bruno., Lorenzo Sciavicco, luigi Villani, Giuseppe Oriolo, "*Robotics: Modelling, Planning and Control*", Springer, 2009.

-- Halaman ini sengaja dikosongkan --

LAMPIRAN

1. Program *Neuro – Fuzzy*

```
function [NFHex] =
NFuzzyHex(xin)
b=0;
%Setpoint Roll dan
pitch '0'
e=xin(1)/120; %
error_roll
de=xin(2)/120; %
error_pitch

%sudut masukkan ke
rotasi matriks
eRoll = xin(1);
ePitch = xin(2);

%koordinat lokal X Y Z
KorK1 = [ 2  3 0 1];
KorK2 = [ 2  0 0 1];
KorK3 = [ 2 -3 0 1];
KorK4 = [-2 -3 0 1];
KorK5 = [-2  0 0 1];
KorK6 = [-2  3 0 1];

%Rotasi Matriks
RnT =
RotTans(eRoll,ePitch,0
,0,0,0);
Rot1 = RnT*(KorK1');
Rot2 = RnT*(KorK2');
Rot3 = RnT*(KorK3');
Rot4 = RnT*(KorK4');
Rot5 = RnT*(KorK5');
Rot6 = RnT*(KorK6');

%learning rate
lr=0.001;

%koordinat tujuan
after Rotasi Roll &
Pitch
Z1 = Rot1(3)
Z2 = Rot2(3)
Z3 = Rot3(3)
Z4 = Rot4(3)
Z5 = Rot5(3)
Z6 = Rot6(3)

%parameter consequent
ck1=0;ck2=0;ck3=0;ck4=
0;ck5=0;ck6=0;
R1=1;
R2=2;
R3=3;
c1(1)=ck1-R3;
c1(2)=ck1-R2;
c1(3)=ck1-R1;
c1(4)=ck1;
c1(5)=ck1+R1;
c1(6)=ck1+R2;
c1(7)=ck1+R3;

c2(1)=ck2-R3;
c2(2)=ck2-R2;
c2(3)=ck2-R1;
c2(4)=ck2;
c2(5)=ck2+R1;
c2(6)=ck2+R2;
c2(7)=ck2+R3;

c3(1)=ck3-R3;
c3(2)=ck3-R2;
c3(3)=ck3-R1;
c3(4)=ck3;
c3(5)=ck3+R1;
c3(6)=ck3+R2;
c3(7)=ck3+R3;
```

```
c4(1)=ck4-R3;
c4(2)=ck4-R2;
c4(3)=ck4-R1;
c4(4)=ck4;
c4(5)=ck4+R1;
c4(6)=ck4+R2;
c4(7)=ck4+R3;

c5(1)=ck5-R3;
c5(2)=ck5-R2;
c5(3)=ck5-R1;
c5(4)=ck5;
c5(5)=ck5+R1;
c5(6)=ck5+R2;
c5(7)=ck5+R3;

c6(1)=ck6-R3;
c6(2)=ck6-R2;
c6(3)=ck6-R1;
c6(4)=ck6;
c6(5)=ck6+R1;
c6(6)=ck6+R2;
c6(7)=ck6+R3;

%inferece mack vicar
wheelan
rbk1 =[6 6 6 7 7;
       4 5 6 6 7;
       2 3 4 5 6;
       4 3 2 2 1;
       1 1 2 2 2];

rbk2 =[3 3 4 5 5;
       2 3 4 5 6;
       2 3 4 5 6;
       6 5 4 3 2;
       3 3 4 5 5];

rbk3 =[1 1 2 2 2;
       1 2 2 3 4;
       2 3 4 5 6;
       7 6 6 5 4;
       6 6 6 7 7];

rbk4 =[2 2 2 1 1;
       4 3 2 2 1;
       6 5 4 3 2;
       4 5 6 6 7;
       7 7 6 6 6];

rbk5 =[5 5 4 3 3;
       6 5 4 3 2;
       6 5 4 3 2;
       2 3 4 5 6;
       5 5 4 3 3];

rbk6 =[7 7 6 6 6;
       7 6 6 5 4;
       6 5 4 3 2;
       1 2 2 3 4;
       2 2 2 1 1];
e1=1;
e2=1;
e3=1;
e4=1;
e5=1;
e6=1;

%menghitung ANFIS
while (e1 >= 0.0001 &&
b < 100000)
ef=zeros(5,1);
def=zeros(5,1);

% lapisan 1
% fuzzifikasi error
roll
   if e<-2
        ef(1)=1;
        %ef(5)=1;
   elseif e<-1
        ef(1)= -1-e;
        ef(2)= e-(-2);
        %ef(5)= -1-e;
        %ef(4)= e-(-
2);
   elseif e<0
```

```
        ef(2)= 0-e;
        ef(3)= e-(-1);
        %ef(4)= 0-e;
        %ef(3)= e-(-
1);
   elseif e<1
        ef(3)= 1-e;
        ef(4)= e-0;
        %ef(3)= 1-e;
        %ef(2)= e-0;
   elseif e<2
        ef(4)= 2-e;
        ef(5)= e-1;
        %ef(2)= 2-e;
        %ef(1)= e-1;
   else
        ef(5)=1;
        %ef(1)=1;
   end
   %ef
% fuzzifikasi error
pitch
    if de<-2
        def(1)=1;
        %def(5)=1;
   elseif de<-1
        def(1)= -1-de;
        def(2)= de-(-
2);
        %def(5)= -1-
de;
        %def(4)= de-(-
2);
   elseif de<0
        def(2)= 0-de;
        def(3)= de-(-
1);
        %def(4)= 0-de;
        %def(3)= de-(-
1);
   elseif de<1
        def(3)= 1-de;
        def(4)= de-0;
        %def(3)= 1-de;
        %def(2)= de-0;
   elseif de<2
        def(4)= 2-de;
        def(5)= de-1;
        %def(2)= 2-de;
        %def(1)= de-1;
   else
        def(5)=1;
        %def(1)=1;
    end
    %def

%lapisan kedua
for i=1:5
    for j=1:5

%w(i,j)=ef(i)*def(j);
%sugeno

w(i,j)=min(ef(i),def(j
)); %mamdani
    end
end

%inference fuzzy mack
vicar wheelan
u1 = zeros(7,1);
u2 = zeros(7,1);
u3 = zeros(7,1);
u4 = zeros(7,1);
u5 = zeros(7,1);
u6 = zeros(7,1);
    for l= 1:5
        for m=1:5
        k = rbk1(l,m);
        %u1(k) =
u1(k)+w(l,m); %sugeno
        u1(k) =
max(u1(k),w(l,m));
%mamdani
        end
    end
    for l= 1:5
        for m=1:5
```

```
        k = rbk2(l,m);
        %u2(k) =
u2(k)+w(l,m);
        u2(k) =
max(u2(k),w(l,m));
        end
    end
    for l= 1:5
        for m=1:5
        k = rbk3(l,m);
        %u3(k) =
u3(k)+w(l,m);
        u3(k) =
max(u3(k),w(l,m));
        end
    end
    for l= 1:5
        for m=1:5
        k = rbk4(l,m);
        %u4(k) =
u4(k)+w(l,m);
        u4(k) =
max(u4(k),w(l,m));
        end
    end
    for l= 1:5
        for m=1:5
        k = rbk5(l,m);
        %u5(k) =
u5(k)+w(l,m);
        u5(k) =
max(u5(k),w(l,m));
        end
    end
    for l= 1:5
        for m=1:5
        k = rbk6(l,m);
        %u6(k) =
u6(k)+w(l,m);
        u6(k) =
max(u6(k),w(l,m));
        end
    end
```

```
%lapisan ketiga
Normalisasi

su1=0;su2=0;su3=0;su4=
0;su5=0;su6=0;

w11=0;w12=0;w13=0;w14=
0;w15=0;w16=0;
 for o=1:7
      su1 = su1 +
u1(o);
      su2 = su2 +
u2(o);
      su3 = su3 +
u3(o);
      su4 = su4 +
u4(o);
      su5 = su5 +
u5(o);
      su6 = su6 +
u6(o);
 end
 for p=1:7

w11(p)=u1(p)/su1;

w12(p)=u2(p)/su2;

w13(p)=u3(p)/su3;

w14(p)=u4(p)/su4;

w15(p)=u5(p)/su5;

w16(p)=u6(p)/su6;
 end

%lapisan keempat
for z=1:7

f1(z)=w11(z)*c1(z);

f2(z)=w12(z)*c2(z);
```

```
f3(z)=w13(z)*c3(z);

f4(z)=w14(z)*c4(z);

f5(z)=w15(z)*c5(z);

f6(z)=w16(z)*c6(z);
end

%lapisan kelima
(defuzzyfikasi)
sigWi1=0;sigWi2=0;sigW
i3=0;sigWi4=0;sigWi5=0
;sigWi6=0;
sigWiO41=0;sigWiO42=0;
sigWiO43=0;sigWiO44=0;
sigWiO45=0;sigWiO46=0;
for t=1:7

sigWi1=sigWi1+w11(t);

sigWi2=sigWi2+w12(t);

sigWi3=sigWi3+w13(t);

sigWi4=sigWi4+w14(t);

sigWi5=sigWi5+w15(t);

sigWi6=sigWi6+w16(t);

sigWiO41=sigWiO41+f1(t
);

sigWiO42=sigWiO42+f2(t
);

sigWiO43=sigWiO43+f3(t
);

sigWiO44=sigWiO44+f4(t
);

sigWiO45=sigWiO45+f5(t
);

sigWiO46=sigWiO46+f6(t
);
end

O51=sigWiO41/sigWi1;
O52=sigWiO42/sigWi2;
O53=sigWiO43/sigWi3;
O54=sigWiO44/sigWi4;
O55=sigWiO45/sigWi5;
O56=sigWiO46/sigWi6;

PP=[ck1 ck2 ck3 ck4
ck5 ck6];
output = [O51 O52 O53
O54 O55 O56]*(-1);
NFHex = output;

%error calculation
%error=(output_real-
output_kontrol)^2/2
e1 = Z1 - O51;
e2 = Z2 - O52;
e3 = Z3 - O53;
e4 = Z4 - O54;
e5 = Z5 - O55;
e6 = Z6 - O56;

%backpropagation
Perubahan nilai
ck1=ck1+lr*e1*1*w11(4)
;
ck2=ck2+lr*e2*1*w12(4)
;
ck3=ck3+lr*e3*1*w13(4)
;
ck4=ck4+lr*e4*1*w14(4)
;
ck5=ck5+lr*e5*1*w15(4)
;
```

```
ck6=ck6+lr*e6*1*w16(4)
;
c1(1)=ck1-R3;
c1(2)=ck1-R2;
c1(3)=ck1-R1;
c1(4)=ck1;
c1(5)=ck1+R1;
c1(6)=ck1+R2;
c1(7)=ck1+R3;

c2(1)=ck2-R3;
c2(2)=ck2-R2;
c2(3)=ck2-R1;
c2(4)=ck2;
c2(5)=ck2+R1;
c2(6)=ck2+R2;
c2(7)=ck2+R3;

c3(1)=ck3-R3;
c3(2)=ck3-R2;
c3(3)=ck3-R1;
c3(4)=ck3;
c3(5)=ck3+R1;
c3(6)=ck3+R2;
c3(7)=ck3+R3;

c4(1)=ck4-R3;
c4(2)=ck4-R2;
c4(3)=ck4-R1;
c4(4)=ck4;
c4(5)=ck4+R1;
c4(6)=ck4+R2;
c4(7)=ck4+R3;

c5(1)=ck5-R3;
c5(2)=ck5-R2;
c5(3)=ck5-R1;
c5(4)=ck5;
c5(5)=ck5+R1;
c5(6)=ck5+R2;
c5(7)=ck5+R3;

c6(1)=ck6-R3;
c6(2)=ck6-R2;
c6(3)=ck6-R1;
c6(4)=ck6;
c6(5)=ck6+R1;
c6(6)=ck6+R2;
c6(7)=ck6+R3;

%nilai C1(1-7)-C6(1-7)
b=b+1;
end
end
```

2. Program Invers Kinematik

```
#define rad       0.0174532f
#define pulseperdeg 10.75275f

#define rad2deg  (180/PI)
#define deg2rad  (PI/180)
#define deg2dat  (3.41) //(1023/300)

#define cx 23
#define fm 76
#define tb 123

float alpha1,beta1,gamma1;
```

```
float pulse1, pulse2, pulse3;

float derajat;
float derajatlama;

void setup() {
Serial.begin(9600);
  yield();
}

void IK(float xx, float yy, float zz){
float r,teta,teta1,teta2,a,b,c,tan_alpha,tan_beta;
float alpha, beta, gamma;
float AA, BB, CC, DD;
float theta1_a,theta2_a,theta3_a,degree1,degree2,degree3;
float buff_1, buff_2, buff_3;
float z, x;
float LL1,LL2,a1,a2,a3;
float a2_1,a2_2,a3_1,a3_2;
float cos_teta2,tan_teta1;
float tan_a,tan_b,tantetta,tetta;

z = zz;
x = xx;
//z = 120 - (zz*(-1));
//x = map(xx,160,0,0,160);
//x = x + 36.5;
//-------------------------------------------------------------------//
r=sqrt((x*x) + (yy*yy));
teta=atan2(yy,x) * 57.29577957855f;;
alpha=rad2deg*atan2(yy,x);

//teta2=acos(((r*r) + (z*z) - (fm*fm) - (tb*tb))/(2*fm*tb))) *
57.29577957855f;
cos_teta2=(r*r) + (z*z) - (fm*fm) - (tb*tb);
cos_teta2/=(2.0f*fm*tb);
teta2=acos(cos_teta2) * 57.29577957855f;
gamma=rad2deg*acos(((r*r) + (z*z) - (fm*fm) - (tb*tb))/(2*fm*tb));
```

```
tan_beta=z/r;
//tan_alpha=tb*sin(teta2*rad)/(fm + (tb*cos(teta2*rad)));
tan_alpha=tb*sin(teta2*rad);
tan_alpha/=fm + (tb*cos(teta2*rad));

tan_teta1=tan_alpha - tan_beta;    //menghasilkan nilai teta2 positif
//tan_teta1=tan_beta - tan_alpha;  //menghasilkan nilai teta2 negatif
tan_teta1/=(1.0f + (tan_beta*tan_alpha));
//Serial.print("tan_teta1:");
//Serial.println(tan_teta1);
teta1=atan(tan_teta1)* 57.29577957855f;
//teta1=atan2((tan_alpha + tan_beta),(1 - (tan_alpha*tan_beta))) *
57.29577957855f;
beta=rad2deg*atan2((tan_alpha - tan_beta),(1.0f +
(tan_alpha*tan_beta)));
//---------------------------------------------------------------------//
AA=((x*x) + (yy*yy) + (z*z) - (fm*fm) - (tb*tb))/(2*fm*tb);
BB=sqrt(1-(AA*AA));
theta3_a=atan2(BB,AA);

buff_1=fm + (tb*cos(theta3_a));
buff_2=sqrt((x*x) + (yy*yy));
buff_3=tb*sin(theta3_a);

CC=(z*buff_1)-(buff_2*buff_3);
DD=(buff_2*buff_1)+(z*buff_3);

theta2_a=atan2(CC,DD);
theta1_a=atan2(yy,x);

tan_a=buff_3/buff_1;
tan_b=z/buff_2;

tantetta=tan_a - tan_b;
tantetta/=(1.0f + (tan_a*tan_b));
tetta=atan(tan_teta1)* 57.29577957855f;

degree1=theta1_a*57.29577957855f;
degree2=theta2_a*57.29577957855f;
```

```
degree3=theta3_a*57.29577957855f;
//------------------------------------------------------------------------//
  alpha1=rad2deg*atan2(yy,xx);
  LL1=sqrt((xx*xx)+(yy*yy));
  LL2=sqrt((LL1-cx)*(LL1-cx)+(zz*zz));
  a1=rad2deg*atan2((LL1-cx),zz);
  //a2=rad2deg*acos(((tb*tb) - (LL2*LL2) - (fm*fm))/(-2*LL2*fm));
  a2_1=((tb*tb) - (LL2*LL2) - (fm*fm))/(-2*LL2*fm);
  a2_2=sqrt(1-(a2_1*a2_1));
  a2=rad2deg*atan2(a2_2,a2_1);
  //a3=rad2deg*acos(((LL2*LL2) - (tb*tb) - (fm*fm))/(-2*tb*fm));
  a3_1=((LL2*LL2) - (tb*tb) - (fm*fm))/(-2*tb*fm);
  a3_2=sqrt(1-(a3_1*a3_1));
  a3=rad2deg*atan2(a3_2,a3_1);
  beta1=90-(a1+a2);
  gamma1=180-a3;

}

void loop() {
  float trayektori();
   trayektori=sqrt((100*100)+(60*60));

   IK(160,0,20);
   delay(3000);

   }
```

-- Halaman ini sengaja dikosongkan --

RIWAYAT HIDUP

Penulis bernama lengkap **Muhammad Fajar Ramadhan** lahir di Sidoarjo, provinsi Jawa Timur pada tanggal 3 Maret 1994. Penulis merupakan anak kedua dari pasangan Bambang Eko Rusdiono dan Try Rahayu. Menamatkan Sekolah Dasar di SDN Tropodo 1 Sidoarjo, kemudian melanjutkan ke SMPN 2 Waru. Untuk jenjang SMA penulis menyelesaikan sekolahnya di SMA Muhammadiyah 3 Surabaya. Setelah menamatkan SMA, penulis melanjutkan pendidikan Diploma III Elektro Computer Control di Institut Teknologi Sepuluh Nopember Surabaya. Setelah menamatkan Diploma III, penulis melanjutkan studi Strata 1 Lintas Jalur di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun 2015. Spesialis bidang studi yang ditekuni oleh penulis adalah Teknik Sistem Pengaturan. Pada bulan Juli 2017 penulis mengikuti seminar dan ujian Tugas Akhir di Bidang Studi Teknik Sistem Pengaturan Jurusan Teknik Elektro ITS Surabaya sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Teknik Elektro. Penulis dapat dihubungi di :

Email: ffajarrzz@gmail.com

-- Halaman ini sengaja dikosongkan --